Tuning of an Aircraft Pitch PID Controller with Reinforcement Learning and Deep Neural Net

Adyasha Mohanty (madyasha@stanford.edu), **Emma Schneider** (epschnei@stanford.edu)

1 Introduction

An aircraft is a highly nonlinear dynamical system that requires control across three different axes- roll, pitch and yaw. Traditionally, this control is achieved using Proportional-Integral-Derivative (PID) controller which also finds usage in nearly 90% of control systems in other domains. PID gains obtained with classical tuning methods exhibit strong static performance but the controller's stability and performance are reduced if the system is exposed to time varying uncertain observations. For an aircraft, such variations may arise due to external disturbances such as wind gusts, uncertainty in aerodynamic characteristics, etc. Using fixed observation-based controllers entails that the gains are optimal only for a specific flight configuration and therefore need to be retuned to function robustly if any stochasticity is added to the environment. In addition to being time-consuming, such methods require exhaustive knowledge of aircraft dynamics and prior information about the desired transient characteristics. This issue can be mitigated by use of an adaptive gain tuning method that produces optimal gains with time-varying and random observations. The goal of this work is to control the longitudinal motion of an aircraft by producing a set of optimal PID gains for every random observation using a combination of Reinforcement Learning(RL) and deep neural net architecture. The gains are then summed as a controller input, represented by the elevator deflection angle to closely track the target pitch value. While the idea of adaptive tuning is not new, its application to an aircraft pitch control in conjunction with a neural net makes the proposed work a novelty. The use of a reward-driven algorithm such as RL relaxes the hard constraint of being fully adept from a control's perspective which is crucial while using a classical PID tuning method that works with fixed observations. The RL agent interacts with an environment repeatedly and tunes the parameters of its policy to maximize the long-term reward using a Deep Deterministic Policy Gradient (DDPG) algorithm. The agent takes a continuous observation vector of the error in pitch angle, derivative and integral of error and produces an output containing a set of three optimal gains, namely the P gain, D gain and I gain for each random observation, thus allowing the controller to function optimally in a wide range of flight configurations and when exposed to disturbances.

2 Related Work

In this work, a RL approach was applied to tune the PID gains to control the speed of a DC motor[i]. The control logic implementation did not rely on the knowledge of the expert as the gains were adjusted based on interactions with the environment. The validation responses showed reduced oscillations and low settling time. However, application of Q-learning is impractical for an aircraft system due to the large number of state-action pairs. Other references perform adaptive tuning for pitch control using fuzzy logic. In every work, the longitudinal equation of motion is decoupled from lateral dynamics and then the model is linearized, consistent with the modeling assumptions made in this project. A fuzzy logic reasoning is applied in this work[ii] in which ~50 rules are designed to maintain the pitch angle of the aircraft. The inputs to the fuzzy algorithm were the pitch error and pitch rates and the outputs were an optimal set of PID gains. The fuzzy controller's response was robust to external disturbances; however, no quantifiable transient characteristics were mentioned which could prove the controller's reliability in the presence of different types of stochastic inputs. Another reference mentions the design of both a traditional PID and a fuzzy logic controller[iii]. The results with the fuzzy controller with 5 membership functions and 17 rules showed fast rise time, settling time and minimal overshoot, although the steady state error was non-zero. The fourth reference[iv] evaluated the performance of a PID controller alongside a self-tuning fuzzy PID. The fuzzy PID resulted in a response with low settling time and a remarkable handling of external disturbances, however the steady state error was shown as non-zero. [v]The last reference also used fuzzy logic and membership functions to damp out oscillations in the pitch response. This paper was successful at reducing the overshoot to 0 % and achieved a rise time of nearly 0.73 seconds with the fuzzy logic implementation, even when the system was exposed to continuous and adaptive disturbances.
The surveys bolster the motivation for the present work that adaptive gains can be applied to a multitude of control systems, including an aircraft pitch controller. However, fuzzy interference logic might not be the only solution for producing such gains. Since the strength of the fuzzy logic is dependent on the knowledge base which consists of the rule base and the data base, this imposes limitations on the types of disturbances the aircraft is exposed to. Additionally, designing fuzzy rules is a trial-and error process and requires expert knowledge. Nevertheless, the promising results show that the application of RL to obtain optimal PID gains would be highly beneficial as the user need not have complete domain knowledge. RL can also be combined in the future with fuzzy logic to achieve a low settling time for the pitch response.

3 Model Design

Six nonlinear coupled differential equations describe an aircraft motion. The pitch dynamics is first decoupled from lateral dynamics and then the equations are linearized based on data from a Boeing commercial aircraft model[vi]. The system input is a desired

pitch angle and the outputs are the pitch angle, pitch rate and angle of attack. Laplace transform of the linearized equations results in a state-space model in SIMULINK as shown in Figure 1.
The RL Agent is designed with MATLAB's RL toolbox[vii]. It takes the observation vector, a reward function and an 'isdone' Boolean. A constant input is randomly selected at each training step from the allowed pitch range for an aircraft, i.e. a pitch angle of -1.5 to 1.5 rad. The reward function is designed in a quadratic manner wherein penalties are applied on the squared control effort and the squared error in the pitch angle, the weights of which were iteratively learned. The 'isdone' block terminates the simulation if the pitch angle exceeds the allowed pitch value. In such cases, the agent gathers no reward at the end of the episode. The output from the RL Agent contains the optimal set of P, I and D gains. Finally, the resulting gains are summed up according to Equation (1) to form a net control effort which is fed into the linearized dynamics model.

$$u = \underbrace{K_p e}_{\text{Proportional Term}} + \underbrace{K_i \int_0^t e\,dt}_{\text{Integral Term}} + \underbrace{K_d \frac{d}{dt} e}_{\text{Differential Term}} \tag{1}$$

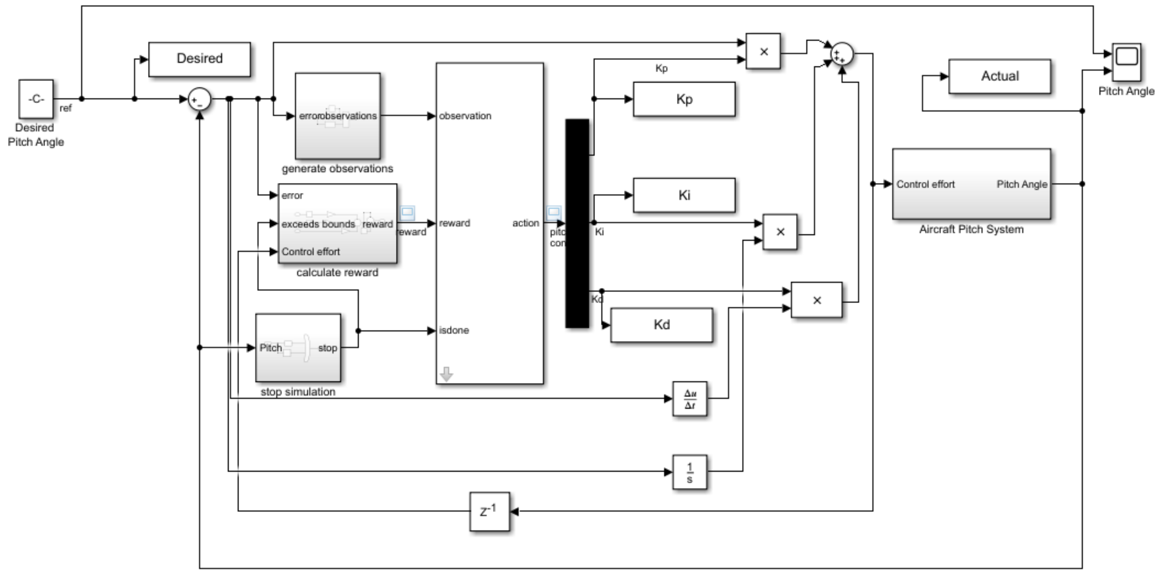

Figure 1: SIMULINK Model for Adaptive Gain Tuning

4 Methods- Neural Net Architecture and Training

DDPG is a model free algorithm that learns policies in high dimensional, continuous action spaces using deep function approximators[viii]. The agent consists of an actor and a critic, each with two fully connected layers, a hidden layer size of 500 and one Rectified Linear Unit (ReLU) activation layer. Figure 2 shows the critic architecture only. A ReLU weight initialization was used for both actor and critic. ReLU activation makes the model less prone to vanishing gradient problem which can make learning time intensive and harder. After the DDPG agent is initialized, an initial action (policy) is computed based on initial observations. The action is propagated in the environment to obtain the next set of observation and reward. The next action is then computed from the experience buffer after adding a perturbation from a stochastic noise model. Next, the current action and the current observation are updated based on a mini batch of experiences randomly sampled from the buffer.

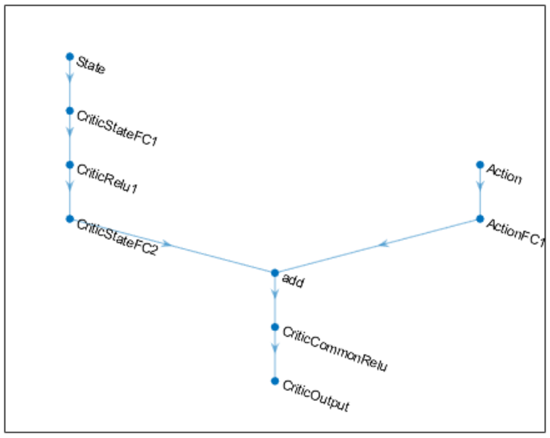

Figure 2: Critic Architecture

The target pitch and actual pitch are visualized with a Scope block connected to the output of the linearized model. If the output pitch angle follows the target pitch value consecutively over a certain number of training episodes, then the training is terminated as the agent can now successfully tune the PID controller which in turn achieves the target response. Validation is performed after an agent has been successfully trained on the model. A set of ten positive and negative pitch values are randomly chosen as constant inputs for the model. The learned gains and actual pitch response are obtained for each validation case. A separate script is utilized to plot the adaptive gains, the target pitch angle, actual pitch angle and to quantify the transient characteristics that determine how well designed a controller is. The transient response is quantified by rise time- time taken by the response to change from 10% to 90% of the target response, overshoot- how much a signal exceeds the desired value, settling time- time taken by the signal to settle to within 2% of its final value and lastly, steady state error, the difference between the settled signal value and the desired signal value. Each gain value has a direct correspondence to each of the transient characteristic. For instance, integral gain helps eliminate steady state error and a derivative gain reduces overshoot and adds damping to the response.

5 Results and Discussion

The equivalence of bias-variance tradeoff was considered by changing the neural network design and the hyperparameters and observing the response curves for twenty validation cases. The values mentioned from now on, represent averaged values over all validation cases and generalized trends.

5.1 Variation in Neural Net Architecture

The number of layers in both actor and critic framework, as well as the number of hidden layer size was varied. Increasing the hidden layer size promoted polynomial exploration, while increasing the number of layers promoted exponential exploration of the environment. Number of layers in actor network was varied from 1 to 8, while the hidden layer size was varied from 50 to 1000. Increasing the number of layers in the actor network to a value greater than 500 lead to overparameterization and oscillation of the discounted long-term reward, while a decrease in the number of layers yielded poor performance. At least 100 neurons were required in each layer of both actor and critic to learn the policy appropriately, however 500 was considered optimal. Every training curve was stopped at an Average Reward value of 999 as at this value, the agent could successfully tune the controller.
Training curves in Figures 3 and 4 show two permutations of no of layers and number of neurons in each hidden layer

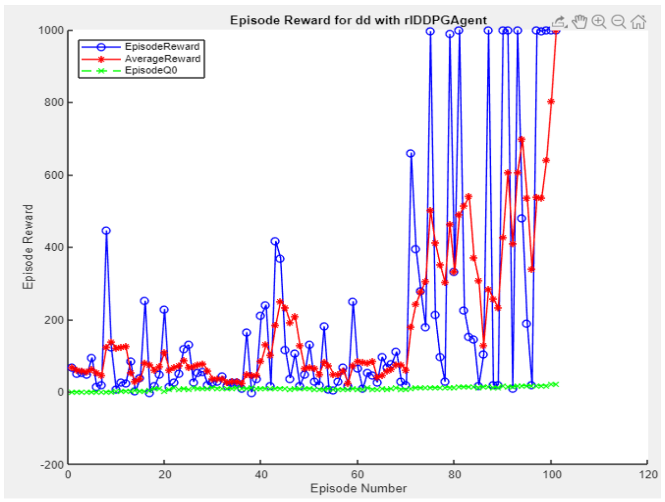

Figure 3: 2 critic layers, 1000 hidden layer size

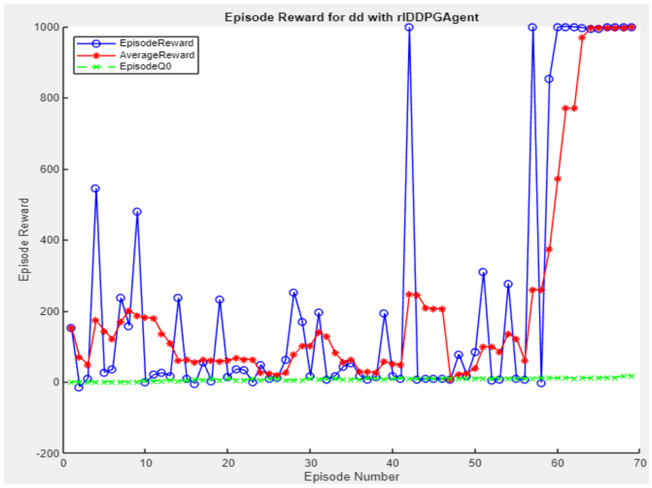

Figure 4: 4 critic layers, 500 hidden layer size

5.2 Variation in Hyperparameters

Training was performed by varying one parameter at a time from a list of available hyperparameters. The optimal hyperparameters are given below:

Table 1: Optimal Hyperparameters

Hyperparameter	Value	Hyperparameter	Value	Hyperparameter	Value
Critic Learn Rate	1e-04	Actor gradient threshold	1	Experience Buffer	1e+06
Actor Learn rate	1e-05	Variance	0.3	Mini-Batch Size	64
Critic Gradient Threshold	1	Variance Decay rate	1e-05	Target Smooth Factor	1e-03

The trends observed are described below, and selected trends are given in Table

- The variance added to the actor was varied from 0 to 0.5, with increments of 0.1. Zero variance (no noise) resulted in the worst response and caused the agent to not converge during training as the agent would not explore the action space at all. Increasing the variance encouraged exploration of action space, however setting this to a large value of ~0.5 resulted in the agent not learning the optimal policy in a timely manner and caused a larger rise time and settling time than desired. Similarly, a small variance of ~ 0.1 led to oscillations, indicating that the agent was caught in a local optimum due to exploitation of a bad policy.

- The critic learn rate was varied from 1e-04 to 1e-05. Further lowering of the learn rate caused slow training and led to greater rise time, settling time and overshoot. The actor learn rate was always set to an order of magnitude lower than the critic learn rate.
- The gradient threshold for both actor and critic were varied among 1,5,10 and infinity. This parameter limits the amount by which gradient parameters are changed during each iteration. Setting a value of infinity resulted in the lowest settling and rise time.
- Mini-batch size was varied among 16, 32, and 256. The last value caused least overshoot but led to a high settling and rise time.

Table 2: Variation in Hyperparameters

Learn Rate (Critic)	**Grad Threshold**		**Variance**	**Mini Batch Size**	**Rise Time (se)**	**Settling Time (sec)**	**Overshoot (%)**	**Steady State Error**
1.00E-04	5	1	0.3	64	1.8996	14.5724	13.6596	0.00005
1.00E-04	inf	1	0.3	64	1.6724	9.6750	21.0580	0.00005
1.00E-04	1	5	0.3	64	2.3835	21.7147	7.6028	0.00010
1.00E-04	1	inf	0.3	64	1.4993	9.4877	22.2037	0.00000
1.00E-04	1	1	0	64	4.9462	69.1554	15.7632	0.00305
1.00E-04	1	1	0.5	64	1.8577	13.4407	15.2806	0.00010
1.00E-04	1	1	0.3	256	1.9249	19.9727	8.4986	0.00045
5.00E-05	5	1	0.3	64	1.3899	12.4596	28.6172	0.00000

5.3 Validation

Although several validation cases were conducted (> 50) on the trained model, the responses for one positive and one negative validation case are shown in Figures 5-8. .

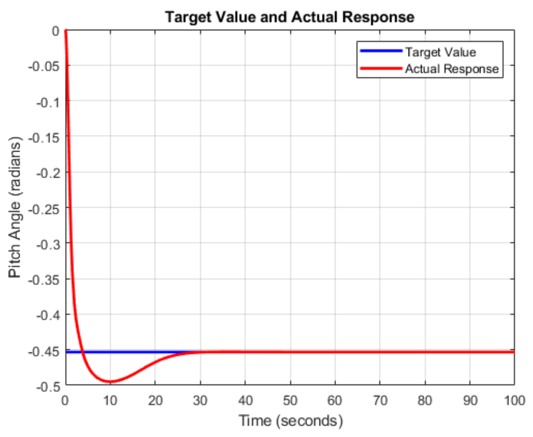

Figure 5: Validation Case-Negative Pitch Angle

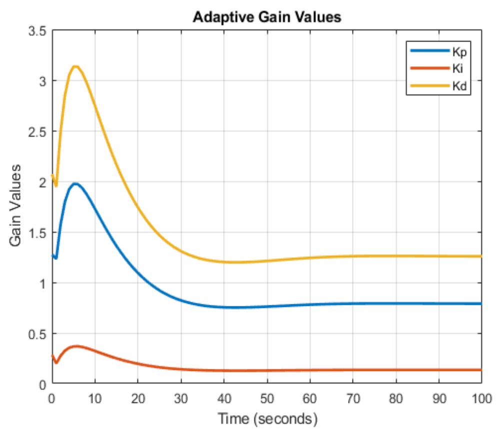

Figure 6: Adaptive gain values-Negative Pitch Angle

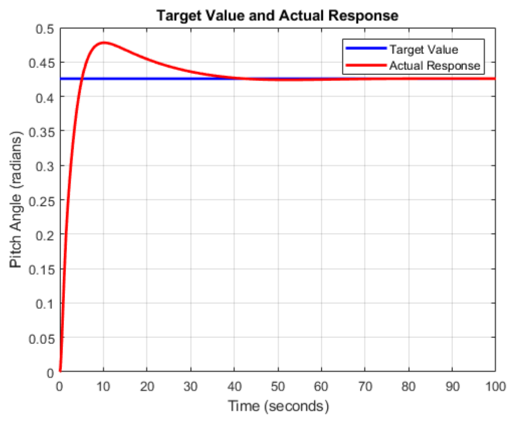

Figure 7: Validation Case-Positive Pitch Angle

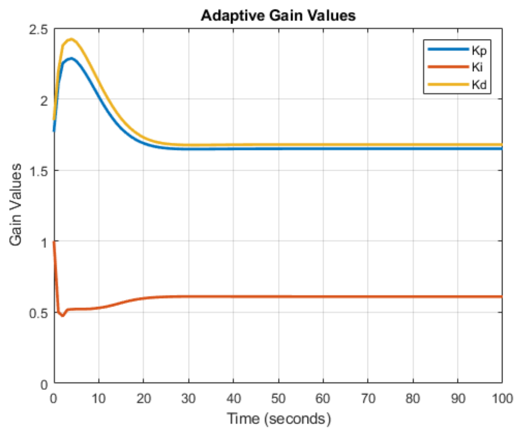

Figure 8: Adaptive gain values-Positive Pitch Angle

The gains are both time and observation dependent, unlike static gains. The adaptively designed controller shows responses that nicely track the target input pitch value, a low overshoot and a fast rise time. However, transient characteristics and comparison with other baseline controllers provide a better insight on the model performance.

A baseline PID controller was designed using SIMULINK's PID controller block and a transfer function-based method was used to tune the controller for a constant step input. The algorithm achieved three objectives: closed loop stability, adequate performance and adequate robustness. The adaptive controller's performance was also compared to two other baseline PID controllers in literature mentioned previously. Although the references also contain responses for fuzzy PID algorithm, the adaptive controller's responses were compared only with the basic PID controllers since the latter represents the case of fixed observations.

Table 3: Comparison of Adaptive Controller to Baseline Controllers

Parameter Name	**Baseline Controller 1 (third tracking point)**[ix]	**Baseline Controller 2 (PID Linear Model)**[x]	**MATLAB PID Tuner App**	**Adaptive Controller**
Rise Time (seconds)	0.2	2.9	6.3836	2.25
Settling Time (seconds)	21.4	6.54	36.9724	24
Overshoot (%)	30.2	14	15.5328	9.5
Steady State Error	0	0	0.0002	0

The transient characteristics obtained with MATLAB's built in PID tuner app clearly perform worse as compared to the adaptive controller, as the rise time and settling time is much higher and the steady state error is non- zero as well. The adaptive controller always produces a zero steady state error regardless of the randomness of the input value. In comparison to baseline controller 1 and 2, the adaptive controller performs better in every aspect except the settling time. However, a key difference to note here is that all the other baseline controllers assume fixed observations, hence the gains reported are always for the specific case that the PID has been tuned for. In the adaptive gains case, however, the numbers reported represent an average over several random desired pitch values, hence, their overall performance is less susceptible to stochasticity in the system. Additionally, the baseline controllers always report lower gain values. This could be a result of repeated tuning by the user and utilization of prior knowledge of the desired transient characteristics. Using the baseline controllers as references, the gains can be made smaller for the adaptive tuning case as well which in turn will reduce the settling time. One such method would be the addition of a hyperbolic tangent layer after the last fully connected layer in the actor network followed by a scaling layer of 0.5 or 0.25, depending on how much the user would desire to lower the gains. However, doing so would implicitly indicate that the user has adequate knowledge of aircraft controls and stabilization and thus would equivalently impose the same restrictions as the other tuning methods. Hence, the gains for the adaptive case were obtained by assuming no prior knowledge of the desired transient response and the desired range of gain values. The only information required prior to training is a linearized model of the system and the pitch range for an operational aircraft. This will allow the user to treat the real controller as a black box and simply train the neural net to track the desired pitch input as closely as possible and output the three optimal gains.

6 Conclusions and Future Work

A trained RL agent was obtained using DDPG algorithm to optimally calculate the PID gains to control the pitch dynamics of an aircraft. This was achieved in three steps-shaping of the reward function with different weights for penalties, varying the neural net architecture with different number of layers and neurons in the hidden layer and obtaining the optimal combination of hyperparameters. The gains were observation varying, hence optimal for every random validation case, thus establishing the robustness of the controller to varying uncertain observations. Validation cases showed that the adaptive controller performs better in terms of transient characteristics such as rise time, steady state error and overshoot when compared to fixed observation based PID controllers found in literature.

Future work involves reduction of the settling time by formulation of new reward functions or training the existing model with other algorithms that work with continuous action spaces such as Proximal Policy Optimization (PPO). The RL agent can also be trained to learn the optimal reward function alongside learning the optimal gains which can drastically reduce the time taken for reward shaping. Additionally, the proposed controller can be tested in an experimental platform such as a RC Aircraft and subjected to artificial wind gusts to evaluate the robustness of the controller. If the flight tests are successful, the algorithm can then be extended to quadcopters for hovering stabilization and obstacle avoidance, although the latter might require more aggressive penalization from the reward function and a much faster rise time for the controller's response.

7 Contributions

Adyasha Mohanty- Model creation, training, experimentation, validation, results and discussion, poster creation
Emma Schneider- Hyperparameter training, Poster and report editing

8 Code

The project files can be accessed at this link: https://github.com/amohanty34/CS-229-Project/tree/master.

9 References

[i] P. Kofinas and A. I. Dounis, "Fuzzy Q-Learning Agent for Online Tuning of PID Controller for DC Motor Speed Control," *Algorithms*, vol. 11, no. 10, p. 148, 2018.

[ii] Torabi, A., Ahari, A., Karsaz, A. and Kazemi, S. (2014). Intelligent Pitch Controller Identification And Design. *Journal of Mathematics and Computer Science*, 08(02), pp.113-127.

[iii] Z. M. Motea, H. Wahid, J. Zahid, S. H. Lwin, and A. M. Hassan, "A Comparative Analysis of Intelligent and PID Controllers for an Aircraft Pitch Control System," *Applications of Modelling and Simulation*, vol. 2, no. 17-25, pp. 1–9, 2018.

[iv] N. Wahid and N. Hassan, "Self-Tuning Fuzzy PID Controller Design for Aircraft Pitch Control," *2012 Third International Conference on Intelligent Systems Modelling and Simulation*, 2012.

[v] N. Beygi and M. Beygi, "Design of Fuzzy self-tuning PID controller for pitch control system of aircraft autopilot," *arXiv-Computer Science-Systems and Control*, Oct. 2015.

[vi] "Aircraft Pitch: System Modeling," *Control Tutorials for MATLAB and Simulink* -. [Online]. Available: http://ctms.engin.umich.edu/CTMS/index.php?example=AircraftPitch§ion. [Accessed: 14-Dec-2019].

[vii] Mathworks, "Train Reinforcement Learning Agents," *Mathworks*, 2019. [Online]. Available: https://www.mathworks.com/help/reinforcement-learning/ug/train-reinforcement-learning-agents.html. [Accessed: 11-Dec-2019]

[viii] . P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous Control With Deep Reinforcement Learning," *International Conference on Learning Representations*, pp. 1–14, 2016.

[ix] Z. M. Motea, H. Wahid, J. Zahid, S. H. Lwin, and A. M. Hassan, "A Comparative Analysis of Intelligent and PID Controllers for an Aircraft Pitch Control System," *Applications of Modelling and Simulation*, vol. 2, no. 17-25, pp. 1–9, 2018.

[x] A. K. Singh, "Design and Modeling of Controllers for Aircraft Pitch Control Movement," *International Journal Of Engineering And Computer Science*, 2016.