COB-2023-0896 COMPARING CLASSIC TO NOVEL FLIGHT CONTROL APPROACHES TO FIXED-WING AIRCRAFT: FEEDBACK CONTROL VERSUS REINFORCEMENT LEARNING

João Erick de Mattos Fernandes
Flávio Luiz Cardoso Ribeiro
Instituto Tecnológico e Aeronáutica
joao.fernandes@ga.ita.br
flaviocr@ita.br

Abstract. *This work aims to compare a classic control method - namely state feedback - to a Deep Q-Network Reinforcement Learning approach applied to the simple problem of altitude holding under disturbance on a longitudinal regional airplane model. A non-linear model was used for simulating the longitudinal behavior of the airplane after applying disturbances to the Angle of Attack and altitude from equilibrium. The disturbances applied were gradually increased in magnitude to compare the reaction of each controller to the point where it can no longer be considered a small disturbance. The same method is repeated with the introduction of a random model uncertainty. Elevator angle was the sole input used for both controllers. The controllers were compared by time domain criteria, such as damp, overshoot, and error as well as how easily those parameters can be directly controlled. A brief review of the Reinforcement Learning method from the control perspective is also presented. Data generated highlights the advantages of using a Reinforcement Learning control - particularly for such problems where the model is unknown, cannot be precisely derived, or changes over time.*

Keywords: *Feedback Control, Fixed-wing Airplane, Reinforcement-Learning*

1. INTRODUCTION

Machine learning tools are considered to have substantial potential for aerospace applications (Brunton et al., 2021). Aircraft control is one of the disciplines that are affected by the development of new tools. Some theoretical application of Machine Learning tools, especially Reinforcement Learning, has been emerging as a research field with successful applications.

On Abbeel *et al* (2007) and Abbeel *et al* (2010), Reinforcement Learning is used to control a helicopter. The first work aims to perform aerobatic maneuvers using Differential Dynamic Programing, while the later one aims to develop autonomous control of the vehicle.

Hwangbo *et al* (2017) used a new algorithm in Reinforcement Learning to stabilize a quadrotor aircraft. The method used could stabilize the vehicle in very extreme initial states, such as manual throwing or upside down. Such stabilizations would be harsh to obtain with classical methods.

In more recent works, Tang and Li (2020) used the Deep Deterministic Policy Gradient (DDPG) to successfully land a fixed-wing aircraft. The same work also explored the reward shaping technique and hyperparameter adjusting, which plays a major role in the controller behavior and performance.

In the work of Clarke and Hwang (2020), a high-level controller was developed to execute aerobatic maneuvers in a fixed-wing aircraft using the Reinforcement Learning method DQN, this method is the same used in this work, as detailed in the next sections.

Zang *et al* (2021) used an RL controller to stabilize a fixed-wing aircraft on cruise using all three control surfaces plus engine thrust control. Along with the other works cited, it is possible to affirm that the use of Reinforcement Learning as a control tool for aerospace purposes is being extensively explored as research.

This work aims to compare this new tool to a classical approach, highlighting the advantages and disadvantages of this new approach in the field of aerospace, specifically in controlling a fixed-wing airplane. The comparison scenario is the altitude stabilization of a fixed-wing aircraft near its equilibrium. The aircraft represented is a fixed-wing airplane modeled after a regional commuter aircraft. A detailed description of the aircraft model and the stabilization parameters are discussed in section 2. The same section encompasses a description of the feedback control technique used as the baseline for comparison with the novel approach to aircraft control. The State Feedback control loop, implemented with the Linear Quadratic method was the tool of choice. In this method, a subset of state variables, regulated by a gain, are used to build a control signal that is fed back to the controlled plant. The gains are adjusted using the maximization of a performance factor, which considers the error of the tracked state and the control power to be used for this tracking.

In section 3, a brief description of Reinforcement Learning is reviewed. This method uses an agent – the controller – to take actions considering, predominantly, the environment. The actions taken by the agent are valued in terms of a reward that the algorithm is designed to maximize. The behavior of the controller is modulated by the precise conception of a reward function. All these concepts are analyzed carefully in section 3, alongside the specificities of the Deep Q-Network (DQN) method (Mnih et al, 2015), which was the method of choice of the present work. From a control perspective, the Reinforcement Learning agent is a non-linear function that maps the observed states to a control signal. This function, for the method chosen, is composed of a neural network. This function is modeled to give results that maximize the reward function in the long term.

Lastly, section 4 aims to confront both methods in qualitative and quantitative terms. Both control approaches are implemented on a deterministic model and a model with uncertainties. The quantitative criteria observed were damp, overshoot, and steady-state error while the qualitative dimensions analyzed were the ease of implementation and adaptiveness to uncertainty.

2. PROBLEM FORMULATION

Both controllers are designed to maintain the desired altitude of a fixed-wing airplane. The airplane is the Generic Narrow Body Airliner – GNBA -, as presented in Guimarães Neto (2014). The following equations of motion are used to describe de airplane dynamics, restricted to the longitudinal motion (Stevens and Lewis, 2003):

$$\begin{aligned}
\dot{V} &= \frac{1}{m}\left(-D + T\cos(\iota_p + \alpha) - mg\sin(\theta - \alpha)\right)\\
\dot{\alpha} &= q + \frac{1}{mV}\left(-L - T\sin(\iota_p + \alpha) + mg\cos(\theta - \alpha)\right)\\
\dot{q} &= \frac{1}{I_{yy}}\left(M + z_p T cos(\iota_p) + x_p T sin(\iota_p)\right)\\
\dot{\theta} &= q\\
\dot{h} &= V sin(\theta - \alpha)\\
\dot{x} &= V cos(\theta - \alpha)
\end{aligned} \tag{1}$$

The state variables are V, the inertial frame velocity; α, the angle of attack; q, the pitch rate; θ, the pitch angle; h, the altitude, and horizontal position x. To obtain these values, the aerodynamic forces L, D, and T (lift, drag, and thrust, respectively) are calculated from the aerodynamic and propulsion models. Variables ι_p and z_p present the orientation of the thrust force application.

For this longitudinal model, three control inputs are available: horizontal stabilizer trim angle, engine thrust (symmetrical), and elevator angle. Both horizontal stabilizer trim angle and engine thrust are used to trim the airplane at a pre-determined desired condition, so the elevator angle is set to zero.

The controllers are designed to act on the elevator angle only. For all the conditions simulated, the horizontal stabilizer trim angle and the engine thrust are kept the same as the trimmed condition. However, it is important to note that due to atmosphere parameters variation with altitude, the thrust value can vary accordingly.

It is also important to note that the system is naturally stable, and the altitude hold value chosen is coincidental to the trimmed altitude.

2.1 Feedback control: PI Output Feedback

For the classical approach, the controller used is a variation of can be found in Stevens and Lewis (2003). The difference of q, theta, and h signals to the equilibrium are fed back to the plant as shown in the diagram below.

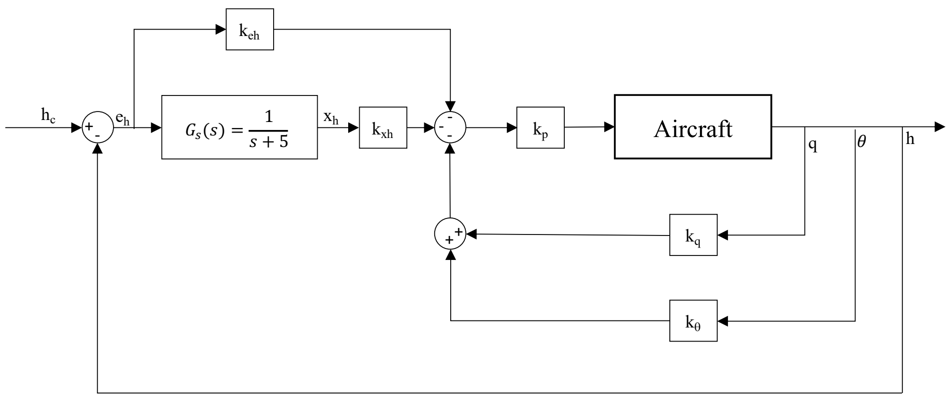

Figure 01. Altitude Autopilot Block Diagram.

The gains, as well as the transfer function shown in the block diagram, could be adjusted manually in an arbitrary way to produce the desired results. However, to optimize the results, a Linear Quadratic technique is used, as explained in the next section.

2.2 LQ Formulation

According to Stevens and Lewis (2003), two concepts are central to modern control system design: the first one is the direct dependency on the state-variable model for the design of the controller. The second is the expression of a "mathematically precise" performance criterion to develop the control gains. The LQ formulation complies with these two concepts, presenting a performance criterion in the following form:

$$J = \frac{1}{2}\int_0^{\infty} (\tilde{x}_a^T Q \tilde{x}_a + \tilde{u}^T R \tilde{u})dt + \frac{1}{2}\bar{e}^T V \bar{e} \tag{2}$$

This equation can be solved using the method of Nelder and Mead (1965), which yields the values of gains that minimize the quadratic function presented. For this equation, $\tilde{x}_a$is the augmented state vector of the closed loop state-space model and $\tilde{u}$ is its respective control input vector. The matrix $\bar{e}$ represents the steady-state error.

For the present work, the values for Q, R, and V were obtained using Bryson method (Bryson Jr. and Ho, 1975), with fine adjustments. The matrix Q, which relates to the regulation level is a semidefinite matrix and is constructed as follows:

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.001 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3}$$

The matrix R relates to the control action level, the greater its value, the less control is used to minimize J. For the present work, R is a one-by-one matrix and has the following value:

$$R = 0.0044 \tag{4}$$

The matrix V relates to the steady state error. For this work it is a one-by-one matrix and is defined:

$$V = 0.0156 \tag{5}$$

The resulting gains enable the closed-loop behavior as presented in section 4.

3. REINFORCEMENT LEARNING

The Markov Decision Process is a framework to describe the learning problem and is one of the fundamental theories for Reinforcement Learning (Sutton and Barto, 2018). In this framework, the interaction between the agent and the environment in terms of states is defined. Reinforcement Learning, then, can be described as a technique to automate learning through the interaction of an agent with the environment with the goal of maximizing the rewards gained through this interaction. To achieve this goal, an RL algorithm does not require supervision and all the learning is granted from the agent-environment interaction.

The Simulink model for the RL controller used for this work is the one that follows.

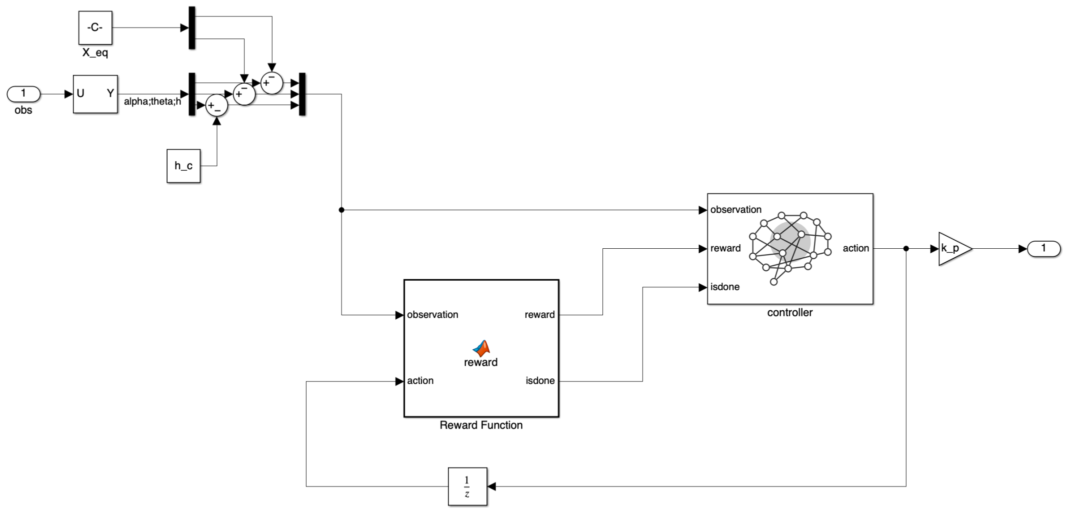

Figure 02. MATLAB/Simulink Reinforcement Learning Controller.

From the Simulink model, it is possible to visualize the formal interaction of the agent (controller) to the environment, as presented in Section 2. A reward function block is also present, signaling to the agent both the reward value and an optional criterion to warn the agent that a critical value is reached, and the simulation is over (the “isdone” variable is set to True and the simulation is halted.) It is also part of the reward block the last action taken, hence the discrete integrator.

A MDP can be described in terms of an agent – the decision maker -, the environment, action, and reward. The next sections comprise an elaboration of these concepts, detailing on how they apply to the present problem as well as the details of the RL method chosen for this work.

3.1 Agent

In general terms, an agent is the core of the reinforcement learning method. According to Sutton and Barto (2018), an “agent must be able to sense the state of its environment to some extent and must be able to take actions that affect the state.”

Despite this simplistic explanation, it is not always trivial to determine the boundary between the agent and the environment. This is because it is possible to design a problem with different levels of complexity and thus the extent and complexity of the agent will follow this composition (Sutton and Barto, 2018). For example, a reinforcement learning agent can be modeled to act on the individual torques of a four-wheeled vehicle and another agent can be built to follow a predetermined trajectory by sending orders to the first one. For the first agent, the slipping of the wheels is part of the environment, since the control of the agent stops at the torque definition, for the second case, it will somehow be considered during the training. The agent-environment boundary is determined by the description of the problem as an MDP.

The heart of the agent is the policy. A policy is how an agent maps an individual state into an action (Sutton and Barto, 2018). The various Reinforcement Learning methods vary on how a policy is implemented, considering all the specificities of the problem.

From a control standpoint, the agent – and especially its policy -, is a non-linear function that maps the observed states into a control signal, producing a desired behavior of the interconnected system. This function should be plastic enough to be progressively tuned during training and the training algorithm should be the most efficient possible to make this function converge to a maximum value of the sum of rewards, given by a reward function.

This framework is not, in its conceptual form, very distant from the classical method presented in Section 2. In this classical method, a value function to be maximized is also present and feedback based on the values of observed states is also clearly the core of the control. The implementation and its results are, nevertheless, very different, and this work aims to bring light to these differences.

For the reinforcement learning part of the present work, the agent is responsible for directly moving the elevator based on the same state variables available for the classical model, with de addition of the last action performed by the agent. For this last additional state, it can be argued that the classical model also takes the control signal into consideration when maximizing the performance index, as discussed in Section 2.

The policy used for the present problem is based on a Critic-based method named Deep Q-Learning. This method is analyzed in the following sections.

3.2 Environment

As could be inferred by previous explanations on agents, the environment can be considered everything external to the agent, or, in other words, everything that is not under direct control by the agent (Sutton and Barto, 2018).

As previously stated, the agent learns the environment through the states. In the present work, the environment is modeled based on the differential equations presented in Section 2. Although not all the state variables are fed to the agent, the ones that are fed are enough for controlling the aircraft efficiently. It is worth noting that different variables are fed to the classical controller and the Reinforcement Learning agent: while the classical approach uses $q, \Delta\theta$ and e_h - altitude error -, the RL controller uses $\Delta\alpha, \Delta\theta$ and e_h.

3.3 Reward

The reward is the formalization of the agent goal (Sutton and Barto, 2018). The computation of a reward takes into consideration the environment and the last action performed. The agent's goal is to maximize the reward for each episode – or simulation session – by choosing the action that maximizes the total reward for the long term, which implies that a low-value action can be chosen in the short term if the total reward is to be maximized.

The reward function is how the control designer interacts with the agent and should be carefully tuned to maximize the desired results. A poorly designed reward function can make possible a local maximum, resulting in an undesired behavior by the controller.

The reward is granted each step – each time an action is performed -, and the total value of the reward is calculated after the episode is over. The simulation can be over by either of two possibilities: by reaching the given final time for the simulation or by reaching the absolute value of 250 meters off the desired altitude. If that happens, the "isdone" variable is triggered, and the simulation stops.

For the agent employed in this work, the reward function is given by:

$$r = 1 + 10\, h_{std} + 10\, \theta_{rwd}; \tag{6}$$

In this formulation, the agent is rewarded by maintaining the episode running for as long as possible, since it is rewarded by an absolute value per step. The variable $\theta_{rwd}$is calculated as follows:

$$\theta_{rwd} = 1 - eh_{abs}/150; \tag{7}$$

The variable eh_{abs} is the absolute value of altitude error. Variable θ_{rwd} is calculated as depicted in Equation 7 only if eh_{abs} is less than 150 and the absolute value of θ is less than 4 eh_{abs}/150+1. Otherwise, the value of θ_{rwd} is zero. With this method of reward, it is possible to quantify the importance of a soft limit to θ, thus avoiding a tendency to overshoot.

The variable h_{std} rewards for maintaining a close value to zero altitude error and is calculated as follows:

$$h_{std} \ = \ 1 - eh_{abs}; \tag{8}$$

The variable h_{std} is calculated as shown on equation 8 when eh_{abs} is less than 1 and the absolute value of θ is 0.75 eh_{abs}. These values express the concept of steady altitude to the controller.

3.4 Action

The action is taken based on a policy. As it should be clear by the goal definition set by the reward, the action is chosen not to collect the highest reward, but to bring about states of high value (Sutton and Barto, 2018). That philosophy will make the collection of the greater sum of rewards possible in the long term.

For the value function, also known as the critic, the DQN method maps the observations in the first layer and outputs, in the last layer, activations for several neurons equal to the number of possible discrete actions in the action space. For this work, the following vector of possible actions is used:

$$\begin{aligned} actInfo \ = \ rlFiniteSetSpec([-25\ -20\ -15\ -10\ -5\ -2\ -1\ -0.5 \\ -0.25\ 0\ 0.25\ 0.5\ 1\ 2\ 5\ 10\ 15\ 20\ 25]); \end{aligned} \tag{9}$$

This vector was built to explore the maximum angle of the elevator and thus the total power of the control, as well as to make possible a fine actuation when it is needed. Therefore, there is a downscaling interval of values as the elevator angle approaches zero. It is important to note, though, that the greater the number of discrete actions, the more complex the neural network becomes. This complexity can, besides demanding more computational power, increase the chance to divergence of the value function, as defined in Mnih (2015).

3.5 DQN Method

The DQN method is an evolution of the Q-learning method and, as its predecessor is a model-free, online, off-policy reinforcement learning method (The Mathworks Inc, 2022). Model-free means this method does not require a model of the environment. Online means it learns directly from the interaction with the environment, rather than a dataset, and off-policy relates to how the agent learns the highest value policy, in this case, it is from a value function rather than the action of the agent itself. While the Q-learning method uses a table for the storage of policy parameters, DQN uses a Deep Neural Network to approximate a value function.

As per observations and actions, the former can be of continuous or discrete kind, while the latter is an action space comprising a finite number of discrete actions. Both the environment and observations are covered in section 2 and the discrete actions are the ones detailed in section 3.4.

The method was first introduced by Mnih *et al* (2013) and refined to make the learning more stable by Mnih *et al* (2015). Two of the tools presented in the later work, the "experience buffer" and the "target value function," are native tools to the DQN method implemented by MATLAB and are used in this work.

According to The Mathworks Inc (2022), during training, considering an ϵ parameter defined as the exploration rate, and a given state S, the agent selects a random action A with probability ϵ. Otherwise $(1-\epsilon)$, the action taken is:

$$A = argmax_A(S, A; \phi) \tag{10}$$

After the action is taken, a new state S' emerges consequently to that action, and the parameters, and the new state S' are stored in an experience buffer. The parameter R is the reward value. A batch is randomly sampled from this experience buffer and the function target y_i is calculated for this batch in the following fashion:

$$y_i = R_i + \gamma max_A Q_t(S_i', A'; \phi_t) \tag{11}$$

The parameter γ ranges from 0 to 1 and corrects the value of farther rewards in time. If this parameter is set to zero, only the last experience counts. Otherwise, if it is set to 1, all the experiences are valued equally, regardless of how far in time the reward was gained.

Having the y_i value resolved, the critic is updated by a one-step optimization of a loss function L across all sampled experiences. The loss function is valued by the following equation:

$$L = \frac{1}{M}\sum_{i=1}^{M}\left(y_i - Q(S_i, A_i; \phi)\right)^2 \tag{12}$$

With the new value of ϕ for the state-action pair, the exploration rate ϵ decays at a given rate, and the process can be restarted. All this process can be explored in more detail by Sutton and Barto (2018).

4. RESULTS AND DISCUSSION

In this section, the simulation results are presented. Four initial conditions were simulated, with a height disturbance from equilibrium ranging from -30 to -120 meters, associated with a variation of the angle of attack ranging from -3 to -12 degrees. The simulation was carried out with a step time of 0.01 seconds, for 30 seconds for section 4.1 and 60 seconds for section 4.2, which was sufficient for the stabilization of the system in both cases.

Section 4.1 presents the results for the deterministic model, while in section 4.2 the simulation results on a model with random variance of aerodynamic derivatives of up to 25% of its original values. Section 4.3 brings the discussion of these results, both in qualitative and quantitative terms.

4.1 Angle of attack and altitude disturbance on the deterministic model

Data generated from a deterministic model enables the comparison in terms of performance for the RL controller and classical controller.

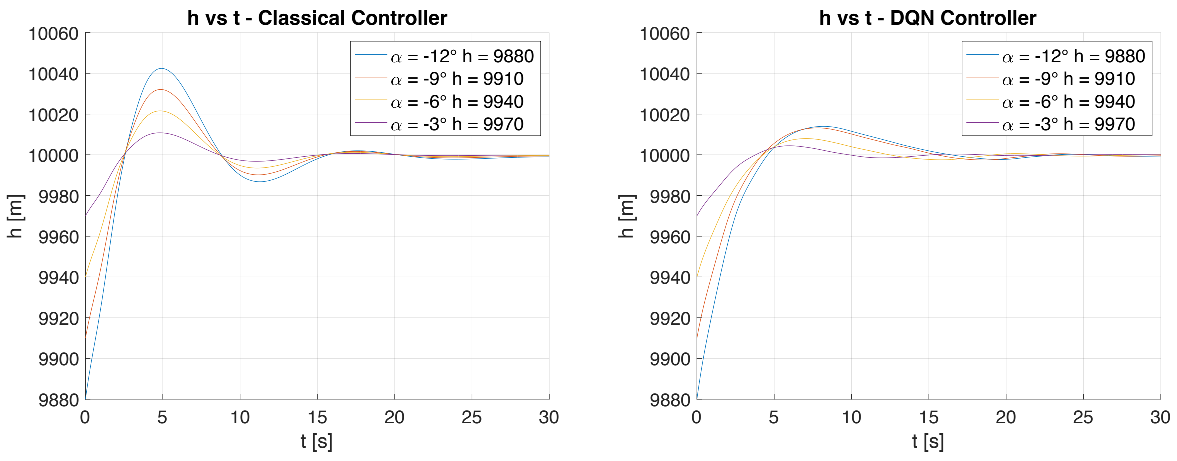

Figure 03. Plot of h *versus* t for various AoA and Height.

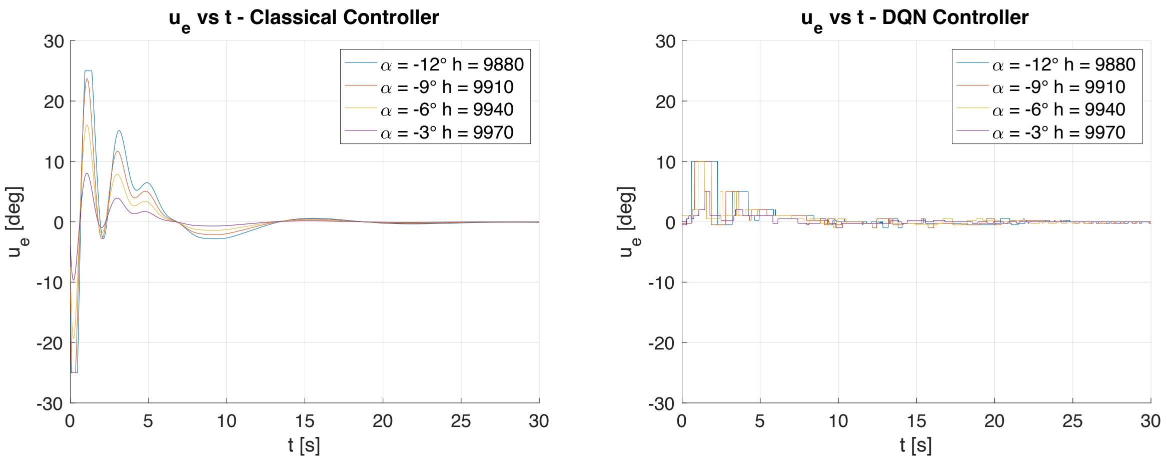

Figure 04. Plot of control signal *versus* t for various AoA and Height.

In Figure 03, it is possible to analyze the behavior relative to the altitude for both controllers. Preliminarily, the difference in frequency is very noticeable. The control input is shown in Figure 04 and the difference in how to controllers react is very evident. In both cases, the proportionality of the control inputs to the altitude error is clear. Nevertheless, the similarities do not go beyond this extent. The RL controller changes control inputs with a lot more frequency and the amplitude also varies.

4.2 Angle of attack and altitude disturbance in the presence of model uncertainty

The same controllers and disturbances are used to compare the controllers in the presence of model uncertainty, introduced by the variation of aerodynamic derivatives.

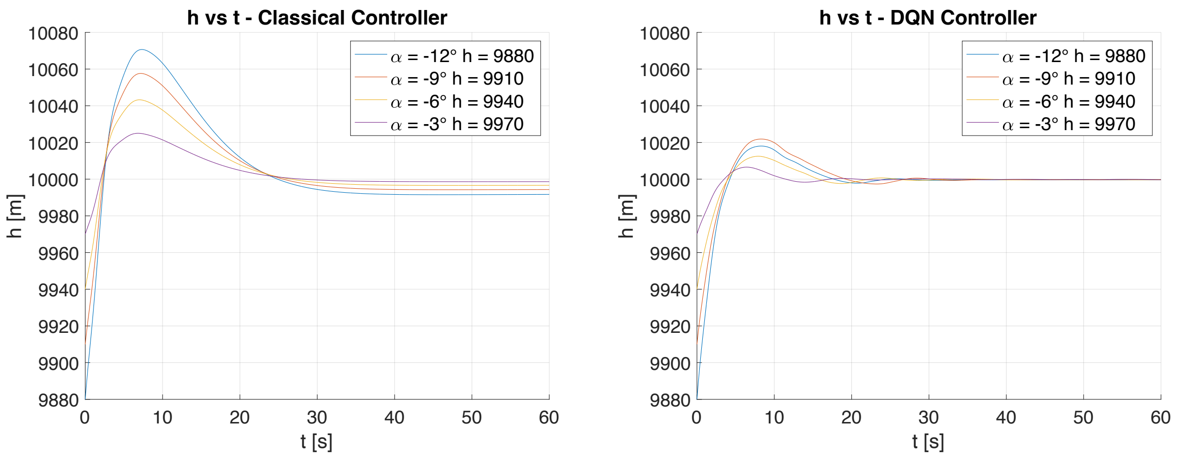

Figure 07. Plot of h *versus* t for various AoA and Height in the presence of uncertainty.

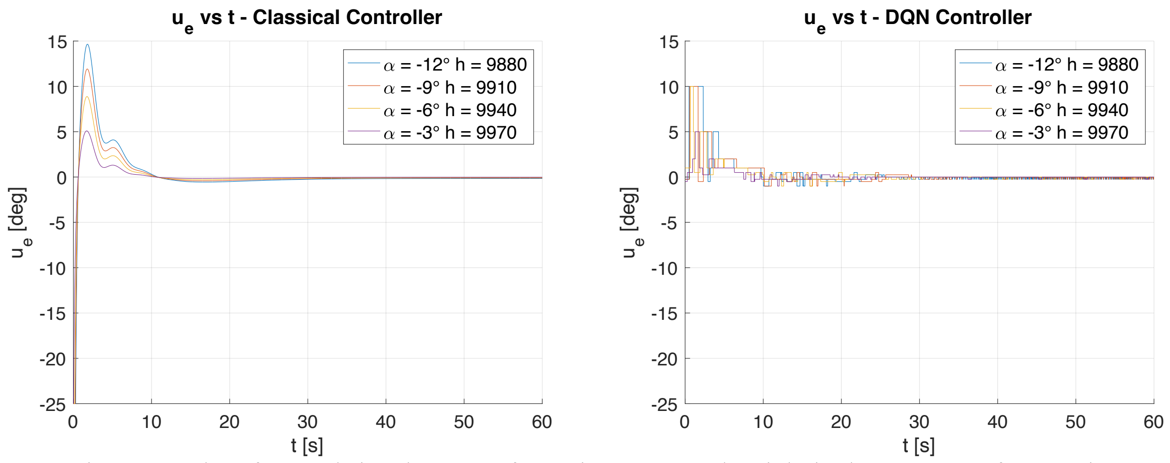

Figure 08. Plot of control signal *versus* t for various AoA and Height in the presence of uncertainty.

Similarly, to the deterministic case, Figure 07 shows an evident disparity of behavior when analyzed in terms of overshoot. For the classical controller, the overshoot increases, which is not observed with the same intensity for the RL controller.

For the control input, as can be seen in Figure 08, the behavior is also like the deterministic case. The RL controller changes control inputs with a lot more frequency and the amplitude also varies in accordance to the altitude error, as expected for both controllers.

4.3 Conclusions and further work

It is possible, from the direct measure of altitude parameter, to compare damp, overshoot, and steady-state error. For the deterministic model, the RL advantage is clear in terms of overshoot. The damp is acceptable for both controllers and the steady-state error is small. The time needed to reach the steady state is also comparable between both controllers, not constituting an appreciable difference in performance.

When uncertainty is introduced in the form of aerodynamic derivatives variation, the RL controller advantage, as expected, becomes clear. While the RL controller retains about the same performance compared to the deterministic model, the same cannot be observed for the classical controller. For the latter, the overshoot increases, and a steady state error proportional to the perturbation is presented.

As per ease of implementation, the engineering choices made by each controller, aside from which state should be fed back to the controlled system, are the ones that impact directly on the control behavior. For the RL controller, the reward function should be chosen carefully not just to express the desired behavior of the controller but also to prevent an optimization of local minima, as could happen if an excessive punishment is set so the agent tends to deliberately end the episode to minimize the losses. For the classical controller, the engineering choices are limited to the Q, R, and V matrices. For both controllers and respective choices, some empirical adjustments are needed.

The classical controller can be easily inspected for stability and robustness. A long list of validated tools is at the disposal of the designer to attest to the quality of the controller in this sense. On the other hand, for the RL controller, it is not as feasible to inspect the control behavior for all the possible states.

The advantage gets reversed when the model's prior knowledge is evaluated. For the classical feedback controller, prior knowledge of the controlled system in terms of a representative model is mandatory since the controller is developed from this model. For the RL controller, no prior knowledge of the model is needed since the controller simply learns the optimum way to interact with it on the basis of the return for a reward function. When in the presence of uncertainty, it is expected that a well-designed RL controller to have a good degree of generalization to adapt to the differences from the training model.

As a recommendation for future works, the same comparison between classical feedback controllers and an RL DDPG (Deep Deterministic Policy Gradient) controller could be explored. Differently from the DQN, widely used in the works referenced, DDPG outputs a continuous control signal that can be constrained into an interval. For this property, it seems to be well-suited to this kind of application.

5. REFERENCES

Abbeel, P.; Coates, A.; Quigley, M.; Ng, A. Y. *An application of reinforcement learning to aerobatic helicopter flight*. Advances in neural information processing systems, 2007, pp. 1–8.

Abbeel, P., Coates, A., and Ng, A. Y., *Autonomous helicopter aerobatics through apprenticeship learning*. The International Journal of Robotics Research, Vol. 29, No. 13, 2010, pp. 1608–1639.

Brunton, S. L.; Nathan Kutz, J.; Manohar, K.; Aravkin, A. Y.; Morgansen, K.; Klemisch, J.; Goebel, N.; Buttrick, J.; Poskin, J.; Blom-Schieber, A. W.; Hogan, T., & McDonald, D. *Data-Driven Aerospace Engineering: Reframing the Industry with Machine Learning*. In AIAA Journal (pp. 1–26). American Institute of Aeronautics and Astronautics (AIAA).

Bryson Jr., A.E.; Ho, Y.C. *Applied optimal control: optimization, estimation, and control*. Washington, DC: Hemisphere, c1975. 481 p. ISBN (10): 0-89116-228-3; (13): 978-0-89116-228-5.

Clarke, S. G., & Hwang, I. *Deep Reinforcement Learning Control for Aerobatic Maneuvering of Agile Fixed-Wing Aircraft*. In AIAA Scitech 2020 Forum. American Institute of Aeronautics and Astronautics.

Guimarães Netto, A. B. *Flight dynamics of flexible aircraft using general body axes: a theoretical and computational study*. PhD Thesis. Graduate Program in Aeronautical Engineering, Instituto Tecnológico de Aeronáutica, São José dos Campos, Brasil.

Hwangbo, J.; Sa, I.,;Siegwart, R.; and Hutter, M. *Control of a Quadrotor With Reinforcement Learning*. IEEE Robotics and Automation Letters, Vol. PP, 2017, pp. 1–1.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. & Riedmiller, M. (2013). *Playing Atari with Deep Reinforcement Learning*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). *Human-level control through deep reinforcement learning*. In Nature (Vol. 518, Issue 7540, pp. 529–533). Springer Science and Business Media LLC.

Nelder, J. A.; Mead, R. *A simplex method for function minimization*. The Computer Journal, v. 7, n. 4, p. 308-313, 1965.

Stevens, B. L.; Lewis, F. L. *Aircraft control and simulation*. 2.ed. Hoboken, NJ: Wiley, c2003.

Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*. 2.ed. Cambridge, Massachusetts. The MIT Press, c2018.

Tang, C.; Lai, Y. -C. *Deep Reinforcement Learning Automatic Landing Control of Fixed-Wing Aircraft Using Deep Deterministic Policy Gradient*. (2020) International Conference on Unmanned Aircraft Systems (ICUAS), Athens, Greece.

The MathWorks Inc. *Reinforcement Learning Toolbox Documentation* (R2022b), Natick, Massachusetts: The MathWorks Inc. https://www.mathworks.com

Zhang, S.; Du, X.; Xiao, J.; Huang, J.; He, K. *Reinforcement Learning Control for 6 DOF Flight of Fixed-Wing Aircraft*. (2021). 33rd Chinese Control and Decision Conference (CCDC), Kunming, China.

6. RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.