




Article

Unmanned Aerial Vehicle Pitch Control Using Deep Reinforcement Learning with Discrete Actions in Wind Tunnel Test

Daichi Wada ^{1,*} , Sergio A. Araujo-Estrada ²  and Shane Windsor ² ¹ Aeronautical Technology Directorate, Japan Aerospace Exploration Agency, Tokyo 181-0015, Japan² Department of Aerospace Engineering, University of Bristol, Bristol BS8 1TR, UK; s.araujoestrada@bristol.ac.uk (S.A.A.-E.); shane.windsor@bristol.ac.uk (S.W.)

* Correspondence: wada.daichi@jaxa.jp

Abstract: Deep reinforcement learning is a promising method for training a nonlinear attitude controller for fixed-wing unmanned aerial vehicles. Until now, proof-of-concept studies have demonstrated successful attitude control in simulation. However, detailed experimental investigations have not yet been conducted. This study applied deep reinforcement learning for one-degree-of-freedom pitch control in wind tunnel tests with the aim of gaining practical understandings of attitude control application. Three controllers with different discrete action choices, that is, elevator angles, were designed. The controllers with larger action rates exhibited better performance in terms of following angle-of-attack commands. The root mean square errors for tracking angle-of-attack commands decreased from 3.42° to 1.99° as the maximum action rate increased from $10^\circ/\text{s}$ to $50^\circ/\text{s}$. The comparison between experimental and simulation results showed that the controller with a smaller action rate experienced the friction effect, and the controllers with larger action rates experienced fluctuating behaviors in elevator maneuvers owing to delay. The investigation of the effect of friction and delay on pitch control highlighted the importance of conducting experiments to understand actual control performances, specifically when the controllers were trained with a low-fidelity model.

Keywords: attitude control; deep reinforcement learning; fixed-wing aircraft; unmanned aerial vehicle; wind tunnel test



Citation: Wada, D.; Araujo-Estrada, S.A.; Windsor, S. Unmanned Aerial Vehicle Pitch Control Using Deep Reinforcement Learning with Discrete Actions in Wind Tunnel Test. *Aerospace* **2021**, *8*, 18. <https://doi.org/10.3390/aerospace8010018>

Received: 16 December 2020

Accepted: 8 January 2021

Published: 14 January 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Many current fixed-wing unmanned aerial vehicles (UAVs) exhibit some nonlinear flight dynamics, such as inertial coupling and aerodynamic nonlinearities. Future aircraft are likely to have even greater nonlinearities in their flight dynamics. For example, bio-inspired morphing wings significantly change their shapes to enhance the agility of UAVs [1,2]. High aspect ratio wings are used in UAVs for high-altitude long-endurance flights [3]. Flexible wings improve the flight efficiency of transport aircraft in off-design conditions [4]. These designs expand the range of applications of UAVs. However, they simultaneously lead to increased nonlinearity, thereby making control challenging. Classical linear controllers generally need to be used in a conservative manner with constrained flight envelopes. As such it is desirable to develop nonlinear control algorithms that can expand the usable flight envelope of UAVs and to take full advantage of the potential of new wing designs.

Machine learning approaches based on neural networks are one type of nonlinear control technique. The nonlinear activation functions in neural networks can represent highly nonlinear transfer from inputs to outputs, which is a promising feature for nonlinear control. A common method for training neural networks is supervised learning, where the neural network is trained using existing labelled data. An example of supervised learning in the aviation domain is the use of a neural network as an alternative to a lookup table in an

aircraft decision making system for collision avoidance [5]. Replacing the numeric tabular system with the neural network increased the efficiency of the system. Neural networks have also been proposed to replace and supplement a number of other components of aircraft flight controller systems [6]. A common approach is to replace the linear gains of conventional control structures with neural networks, and with the performance benefits of this approach having been validated through simulation [7]. Neural networks with [8] and without [9] recurrent architectures have been also used to represent nonlinear inverse transformations for feedback linearization using supervised learning.

A challenge in supervised learning is that training data must accurately reflect the underlying dynamics of a system based on the prior knowledge of the system. In this regard, deep reinforcement learning is a promising alternative to supervised learning. In deep reinforcement learning, agents with neural networks interact with environments. Based on the experience collected through repetitive interactions, the neural networks are updated such that the agents can solve a task, which refers to the development of an optimum control law for aircraft attitude control in this case. The desirable behavior of agents is designed using reward functions, and the agents learn how to maximize the expected reward. In other words, without explicit knowledge of the system, a controller learns to be as nonlinear, accurate and robust as required to solve the task. The powerful task solving capabilities that are difficult to achieve via other methods have been demonstrated specifically in the game playing area [10,11]. Examples of recent deep reinforcement learning algorithms are deep deterministic policy gradient (DDPG) [12], normalized advantage functions (NAF) [13], asynchronous advantage actor-critic (A3C) [14], trust region policy optimization (TRPO) [15] and proximal policy optimization (PPO) [16]. In the aviation domain, reinforcement learning has been applied to glider soaring [17]. A flight policy was successfully learnt to effectively gain lift from ascending thermal plumes. For deep reinforcement learning applications to the attitude control, neural-network-based controllers trained with DDPG, TRPO, and PPO have been tested using simulation for a quadrotor [18]. The controller trained with PPO outperformed conventional proportional–integral–derivative control systems in terms of accuracy and agility. For fixed-wing UAVs, NAF has been applied to learn aerobatic maneuvers [19], and PPO has been applied to control attitudes, including roll, pitch, and airspeed [20]. In addition to flight control, deep reinforcement learning has also been applied to active flow control to stabilize lift and drag fluctuations and to reduce drag [21,22].

The above results have demonstrated the applicability of deep reinforcement learning. Nevertheless, the application of deep reinforcement learning in the aviation domain and its role in attitude control have not been extensively investigated yet. Specifically, most previous works are proof-of-concept studies based on simulation. In practice, neural networks are trained through simulation. However, this method is susceptible to model parameter uncertainty. The theoretical performance is not valid if the simulator used in training is significantly different from the actual operating environment. This is referred to as the reality gap. Therefore, it is critical to assess performance in a real-world situation via experiments to investigate the effect of reality gaps on attitude control.

Considering this background, this study focused on the experimental application of deep reinforcement learning to aircraft attitude control in order to gain understandings of its applicability. To investigate the performance of deep reinforcement learning with a simple and clear basis, wind tunnel tests were conducted using a free-to-pitch one-degree-of-freedom model. Three neural-network-based controllers with discrete action spaces were trained using the A3C algorithm [14]. The controllers output elevator angle rates as actions. The amplitudes of the actions varied between the three controllers. The pitch control performance of the controllers was investigated and compared through simulation and experiments. In particular, the effects of delay and friction on pitch control were investigated as examples of the reality gap.

2. Methods

2.1. Aircraft Model

The one-degree-of-freedom model used in this work is depicted in Figure 1. The model was supported by a shaft, which allowed for free-to-pitch motion. The model was the span-wise half of an off-the-shelf radio control aircraft (WOT4 Foam-E Mk2+, Ripmax, Hertfordshire, UK), which was mounted to the wind tunnel wall. The wing shape was rectangular with a half span of 0.60 m and a chord of 0.25 m. The model was equipped with an inertial measurement unit (IMU) (in Pixhawk 1, 3DR) for measuring the pitch angle and pitch rate, an air speed sensor (custom-built based on SDP31, Sensirion, Stäfa, Switzerland) for measuring the wind speed, and a servo motor for the elevator maneuver. These sensors and the servo were connected to a micro-controller unit. The pitch angle was considered equal to the angle of attack. Measured data were received at the micro-controller unit and transmitted to a computer via USB communication. The computer collected and recorded the data and computed elevator commands, which were transmitted to the elevator servo via the micro-controller unit. Data were processed at 20 Hz control rate, which was also applied in numerical simulations. The implementation of the deep reinforcement learning control algorithm was divided into offline training and online closed-loop control. The offline training was performed via simulations in Matlab using a custom implementation of the A3C algorithm. A computer running the Microsoft Windows Operating System was used for the online closed-loop control. Note that the trained model was run in Simulink, the micro-controller unit was used to collect sensor data and to control the elevator servo, while a custom Python algorithm was used to handle data exchange between Simulink and the micro-controller unit. A simple Network Time Protocol was used to reduce communication latency between the computer and micro-controller unit, with the computer providing the reference clock signal. Additionally, a cooperative task scheduler was used in the micro-controller unit to ensure all tasks run on time.

Figure 2 shows the angle of attack and pitch rate responses when a scheduled elevator angle sweep was applied. The elevator angle was actuated within $\pm 45^\circ$. The experimental results are indicated in black. The angle-of-attack amplitude decreased as the frequency of the elevator sweep increased. The system cut-off frequency was estimated to be 1.3 Hz. A linear model was used to approximate the model dynamics, which was expressed as

$$I_{yy}\ddot{\alpha} = \frac{1}{2}\rho V^2 S c C_M, \quad (1)$$

$$C_M = C_{M_0} + C_{M_\alpha}\alpha + C_{M_q}\frac{c}{2V}q + C_{M_{\delta_e}}\delta_e, \quad (2)$$

where I_{yy} is the moment of inertia for pitch, ρ is the air density, V is the air speed, S is the wing area, c is the chord length, α is the angle of attack, q is the pitch rate, δ_e is the elevator angle, and C represents individual coefficients. The blue plots in Figure 2 indicate the simulated model responses that were obtained when the parameters were set as shown in Table 1. At higher frequencies of the elevator sweep, the simulated responses of the angle of attack and pitch rate were smaller than the experimental results. In addition, the experimental angle of attack showed a delay compared to the simulation response when the elevator started to move at 3 s. This was due to friction, which existed around the pitching shaft. Based on the friction model proposed by Makkar et al. [23] and assuming only the Coulomb friction effect, friction was included in Equation (1) as

$$I_{yy}\ddot{\alpha} = \frac{1}{2}\rho V^2 S c C_M - g_4 \tanh(g_5 q), \quad (3)$$

where the coefficients were estimated as $g_4 = 8.53 \times 10^{-1}$ Nm and $g_5 = 100.0$ s/rad. The red plots in Figure 2 indicate the responses when the model included the friction. The simulated responses and experimental results were still different at higher frequencies of the elevator sweep. However, the simulated responses obtained using the model with friction agreed more closely with the experimental results compared to the simulated responses obtained using the model without friction.

This study employed the linear model without friction as a simulator in the reinforcement learning. This implies that the neural networks were trained with a low-fidelity simulator, that is, there was a reality gap. To highlight the reality gap, Section 4 presents the investigation of the effect of friction on pitch control through numerical simulations.

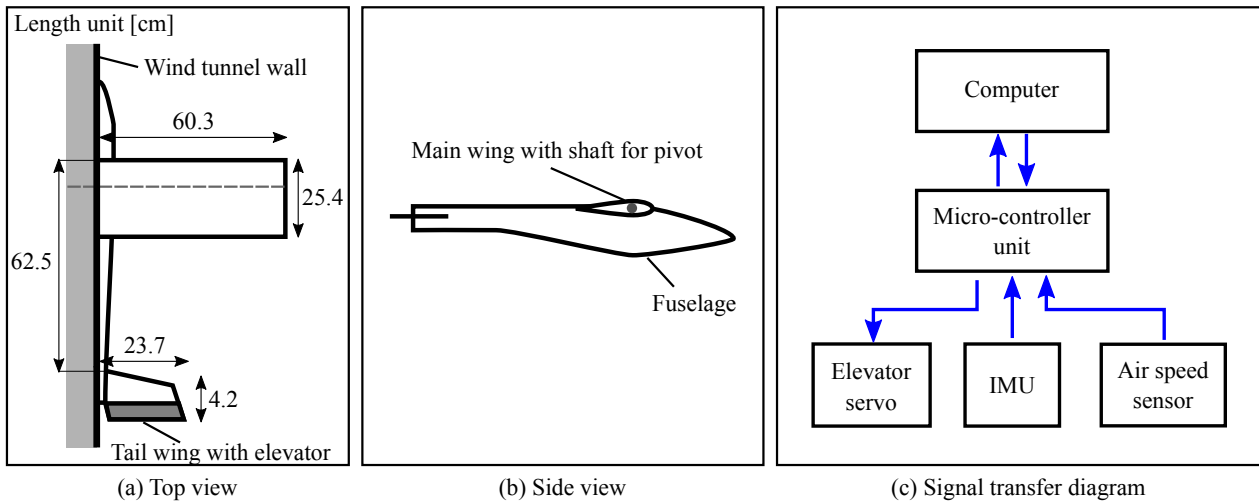


Figure 1. Schematics of the experimental configuration.

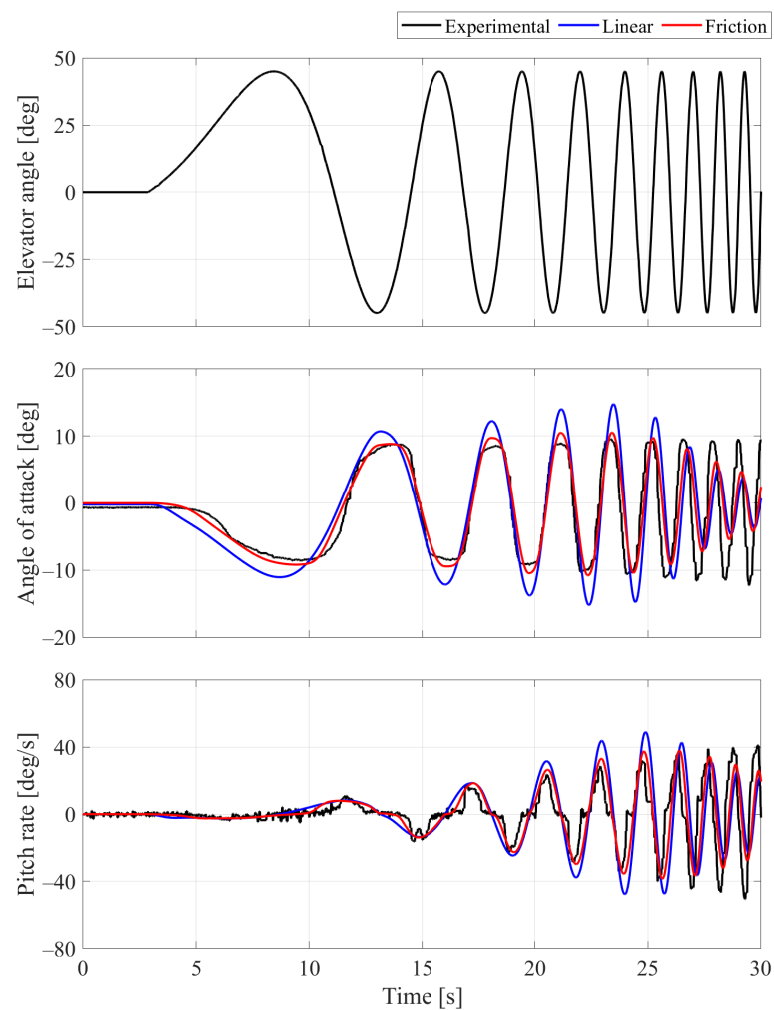


Figure 2. Elevator command (**top**) and related angle of attack (**middle**) and pitch rate (**bottom**) responses at a wind speed of 14 m/s.

Table 1. Parameters of wind tunnel model.

Parameter	Units	Value
I_{yy}	kg m ²	1.90×10^{-1}
ρ	kg/m ³	1.23
V	m/s	14.0
S	m ²	3.06×10^{-1}
c	m	2.54×10^{-1}
C_{M_0}	–	-3.00×10^{-3}
C_{M_α}	–	-2.25×10^{-1}
C_{M_q}	–	-5.46
$C_{M_{\delta_e}}$	–	-5.35×10^{-2}

2.2. Training Algorithm

A deep reinforcement learning algorithm based on A3C was employed [14]. A3C is a model-free and actor-critic method. For policy (actor network) update, it uses a value function (critic network) to reduce gradient variance. It also uses advantage, which is estimated by subtracting action-value from state-value, to assist reducing the variance of the gradient estimation. Since it has a policy-based aspect, it is applicable to both discrete and continuous action spaces.

Agents have a policy function and a value function, and they interact with the environment. The agents observe the state of the environment, perform an action based on the policy to update the environment, and consequently, receive a reward. The two functions are represented by neural networks. The policy function is updated in a gradient ascent manner based on the estimated advantage of performing the action, and the value function is updated in a supervised manner to estimate the advantage based on the collected rewards. Through these updates, the agents learn to control the aircraft model to maximize the expected reward.

In the pitch control tasks, angle-of-attack command schedules were provided to the agents. Wind speeds varied for individual test cases. The agents observed the state, which consisted of the error between the commanded and observed angles of attack, $e_\alpha = \alpha_{com} - \alpha$, observed angle of attack α , pitch rate q , elevator angle δ_e , and wind speed V . The agents fed the state into their neural networks to compute an action, which was the elevator angle rate, to follow the angle-of-attack commands. The neural network architecture used in this study is shown in Figure 3. The architecture was empirically designed. The input layer contained five neurons that corresponded to the state elements. The hidden layer consisted of 16 neurons with sigmoid activation functions. The softmax and linear activation functions were used for policy and value outputs, respectively. The neural network with weights θ considered state s_t at time step t as an input, and it output the action values for possible actions a_t . This was the policy function, $\pi(a_t|s_t; \theta)$. In addition, the neural network output the value of the state $V(s_t; \theta_v)$, where θ_v denotes the weights and biases for calculating the value. The policy and value functions shared all parameters except for the output layers. All weights and biases were initialized based on the Nguyen–Widrow algorithm [24].

Three neural networks with different discrete action choices for elevator angle rate were designed. Elevator angle rate rather than elevator angle was used as the output in order that the whole range of elevator angle was continuously covered by the discrete actions. The NN10 neural network contained three action choices for the elevator angle rate, that is, -10 , 0 , and $10^\circ/\text{s}$. The NN30 neural network contained five action choices, that is, -30 , -10 , 0 , 10 , and $30^\circ/\text{s}$. The NN50 neural network contained seven action choices, i.e., -50 , -30 , -10 , 0 , 10 , 30 , and $50^\circ/\text{s}$. At each time step during simulation, the elevator angle was incremented at the selected rate. The working range of the elevator angle was within $\pm 45^\circ$. The elevator angle was saturated when an action choice attempted to increment it out of the working range.

In training, the duration of a single episode was 30 s with 600 time steps, assuming a control rate of 20 Hz. In each episode, a constant wind speed was randomly selected

within the range of 8 to 22 m/s. The initial states were set as $e_\alpha = 0$ and $\delta_e = 0$. Pitch rate q was set randomly within the range of -5 to $5^\circ/\text{s}$ as an initial perturbation. The angle-of-attack commands were set for every episode using different random seeds. The command schedules were generated by utilizing the Ornstein–Uhlenbeck process [25] with mean $\mu = 0$, volatility $\sigma = 2$ and reverting rate towards the mean $\theta = 0.1$. This process generates a signal that drifts in the same direction for a longer duration rather than oscillating around the mean value. The signal was calculated for 15 points over 30 s. The initial commands were randomly set within the range of -0.5° to 0.5° . The command values were interpolated between adjacent signal points, which resulted in a linear transition of 2.14 s for every interval. Figure 4 shows the examples of the angle-of-attack command schedules. The circles represent the signals calculated by the process. The command schedules varied approximately from -10° to 10° . ϵ -greedy exploration was employed at each time step. A random action was sampled from all choices with the probability defined by ϵ ; otherwise, the action with the maximum action value was sampled. The value of ϵ started at 0.40 and linearly decreased to a minimum of 0.15 after 5000 episodes.

The following loss function was defined to update the weights and biases of the neural network:

$$L = L_\pi + 0.5L_v + 0.01L_{reg}, \quad (4)$$

where L_π is the policy loss, L_v is the value loss, and L_{reg} is the regularization loss with policy entropy. The coefficients for the policy and regularization losses were empirically determined. Following the implementation of the A3C algorithm, the value function should converge to an estimation based on the rewards in the n -step forward view as [14]:

$$V(s_t; \theta_v) = \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n V(s_{t+n}; \theta_v), \quad (5)$$

where r_{t+i} is a reward given at time step $t+i$ and γ is a discount factor. Therefore, the value loss is calculated as

$$L_v = \left(\sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n V(s_{t+n}; \theta_v) - V(s_t; \theta_v) \right). \quad (6)$$

In this work, n was set as 4 for the n -step forward approach. The policy should maximize the expected value; thus the policy loss is calculated as

$$L_\pi = -\log \pi(a_t | s_t; \theta') A(s_t, a_t; \theta, \theta_v), \quad (7)$$

where the gradients are calculated for θ' . $A(s_t, a_t; \theta, \theta_v)$ is an estimate of the advantage function, which is defined as [14]:

$$A(s_t, a_t; \theta, \theta_v) = \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n V(s_{t+n}; \theta_v) - V(s_t; \theta_v). \quad (8)$$

Intuitively, it is an estimate of the advantage of performing action a_t in state s_t based on the policy, $\pi(a_t | s_t; \theta)$.

The regularization loss with policy entropy was added to the loss function to encourage exploration in action. The regularization loss is calculated as

$$L_{reg} = -H(\pi(\cdot | s_t; \theta')), \quad (9)$$

where H is the entropy, which is expressed as

$$H(\pi(\cdot | s_t; \theta')) = -\sum_{a_t} \pi(a_t | s_t; \theta') \log \pi(a_t | s_t; \theta'). \quad (10)$$

The immediate reward was provided at each time step. The reward function was designed based on the angle-of-attack error, e_α , and the action penalty as

$$r = C_{r0} - |e_\alpha| - C_{act}|a_t|, \tag{11}$$

where $C_{r0} = 2 \times \alpha_{lim}$ is a positive constant that is used to maintain the reward value as positive. α_{lim} indicates the working range of the angle of attack, which was set as 15° . If the observed angle of attack, α , exceeded $\pm\alpha_{lim}$, the episode was terminated with $r = 0$. The reward included an action penalty to discourage excess maneuvers. The action penalty was defined based on the output of the neural network, that is, the elevator angle rate. The amplitude of the action penalty, C_{act} , was empirically determined as 0.001 for training NN10 and NN30 and 0.0003 for training NN50.

Multiple agents can run asynchronously using the A3C algorithm. Three agents with different random seeds, and hence different angle-of-attack command schedules and wind speeds, ran in different threads to collect training data, which consisted of states and rewards. The agents sent the training data to global storage. An optimizer that ran in a different thread used the data to update the global neural network, which was then broadcast to the local agents. The Adam optimization algorithm was used as an optimizer, and the hyperparameters were set based on a previous study [26]. The learning rate was 0.001. The optimization was performed whenever the optimizer thread was available. Eventually, a dataset of approximately 400 time steps was used for every update.

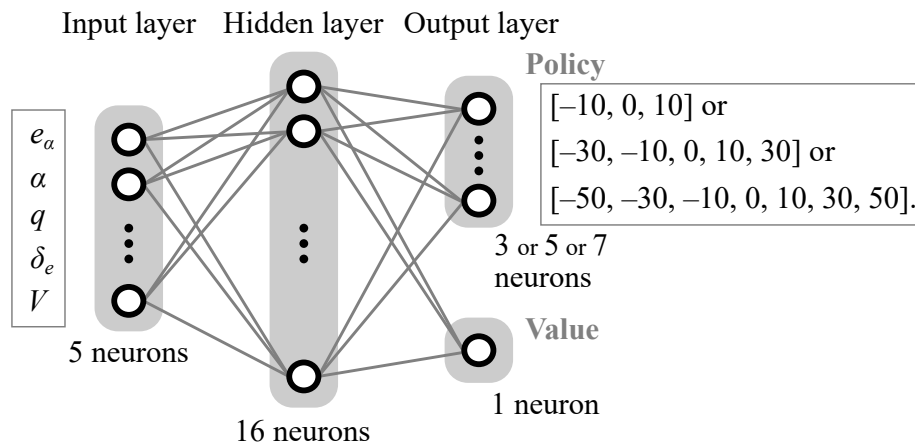


Figure 3. Neural network architecture.

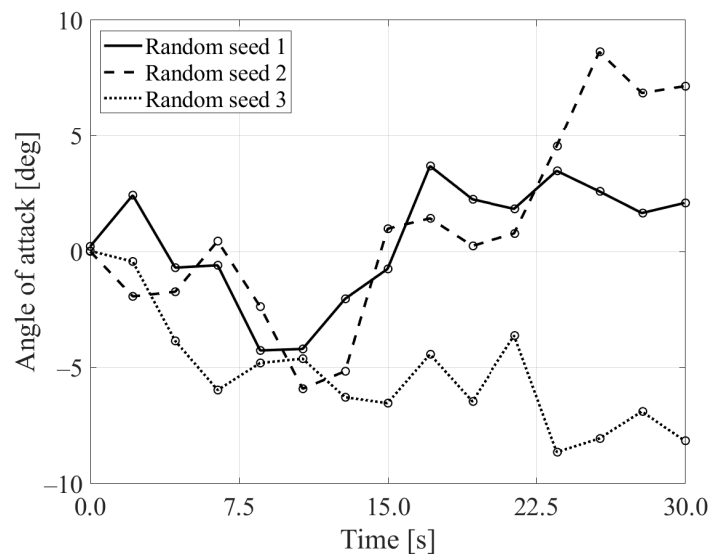


Figure 4. Examples of the angle-of-attack command schedules using different random seeds.

2.3. Experimental Conditions

The wind tunnel consisted of a test section with dimensions of 2.13 m × 1.52 m. Wind speeds were set to be constant during each test case. The applied wind speed range was 10 m/s to 20 m/s with increments of 2 m/s. Doublet and sinusoidal waves with an amplitude of 5° were applied for the angle-of-attack commands. The duration of the doublet wave was 7.5 s at ±5° and steady states. The sinusoidal wave had a frequency of 0.2 Hz and a duration of four cycles. The angle-of-attack commands in the training had neither the smooth trajectories of sinusoidal waves nor the instantaneous transitions of doublet waves. Therefore, the doublet and sinusoidal waves were expected to highlight the generalized performance of the controllers.

There existed a time delay in the experimental configuration. Tests carried out to find the reason behind this delay indicate that the main contributor was the use of a computer as the source of the reference clock signal. The red plot in Figure 5 shows the histogram of the measured communication delay between the controller output and the arrival at the micro-controller unit, which was connected to the elevator servo. The vertical axis represents the relative probability, which is normalized such that the sum of the bar heights is 1, with 15,000 data points being sampled. The peak occurred at 15 ms. A theoretical model was built to simulate delay using a log-normal probability density function, $f(x)$, as

$$f(x) = \frac{1}{(x - t_0)\sigma\sqrt{2\pi}} \exp\left(-\frac{(\log(x - t_0) - \mu)^2}{2\sigma^2}\right), \text{ for } x > t_0, \quad (12)$$

where x is time in milliseconds, t_0 is offset, σ is 1.67 and μ is -5.27 . The blue plot in Figure 5 shows the simulated delay when $t_0 = 15$ in milliseconds. In addition to this type of delay, there are mechanical delays such as those caused by the elastic deformation of the servo horns and links. The total effective delay, which includes communication and mechanical delays, affects pitch control performance. However, the total effective delay was not directly measurable.

In this study, delay was not considered in training. The effect of delay on pitch control is presented in Section 4.

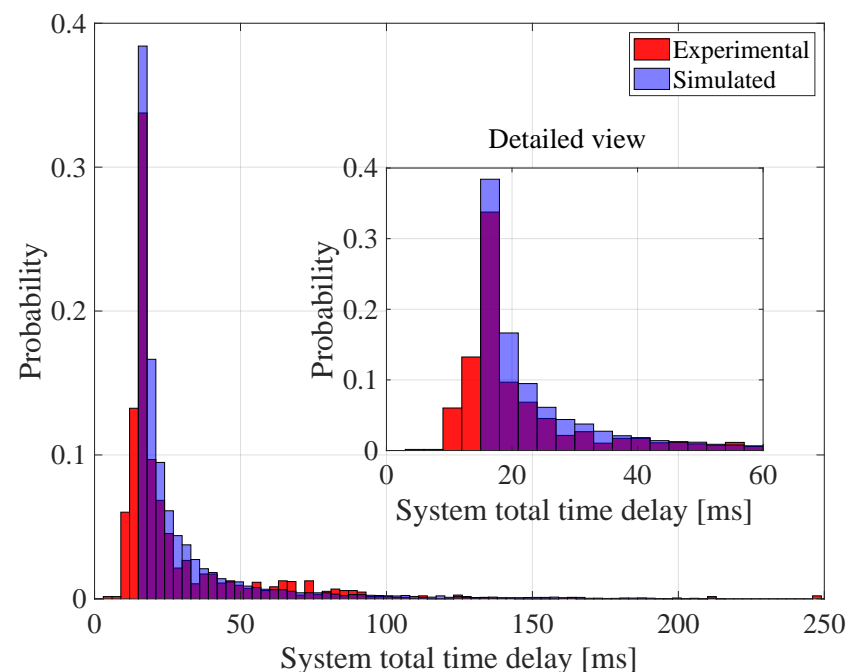


Figure 5. Histogram of command transmission delay. Delay was measured between the controller output and the arrival at the micro-controller unit.

3. Results

Figure 6 shows the results of training. The total rewards were normalized by $1/(600 \times C_{r0})$, where 600 is the number of time steps per episode. All curves were saturated close to 1.0. This indicated that the training successfully converged.

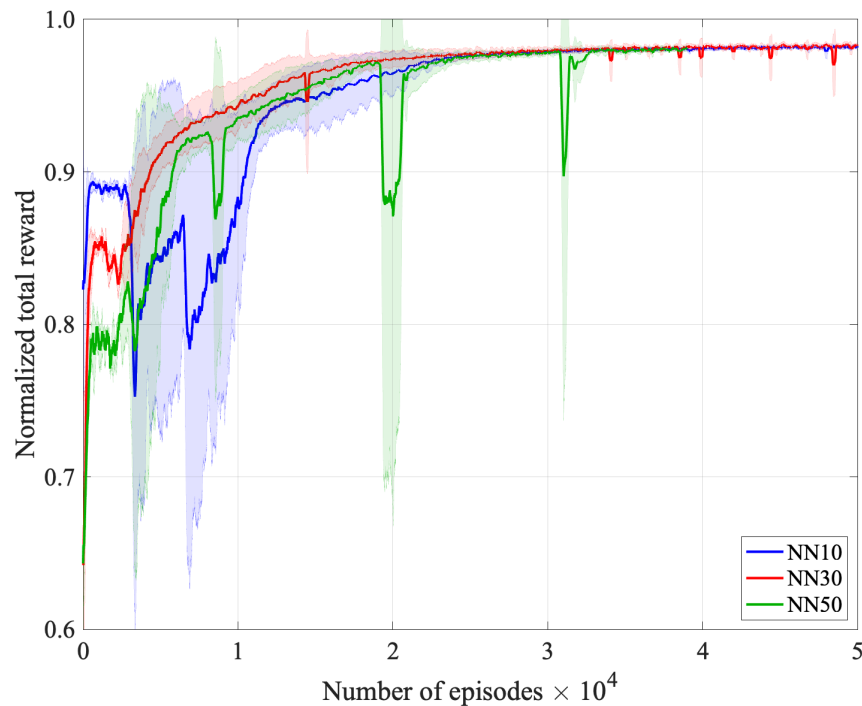
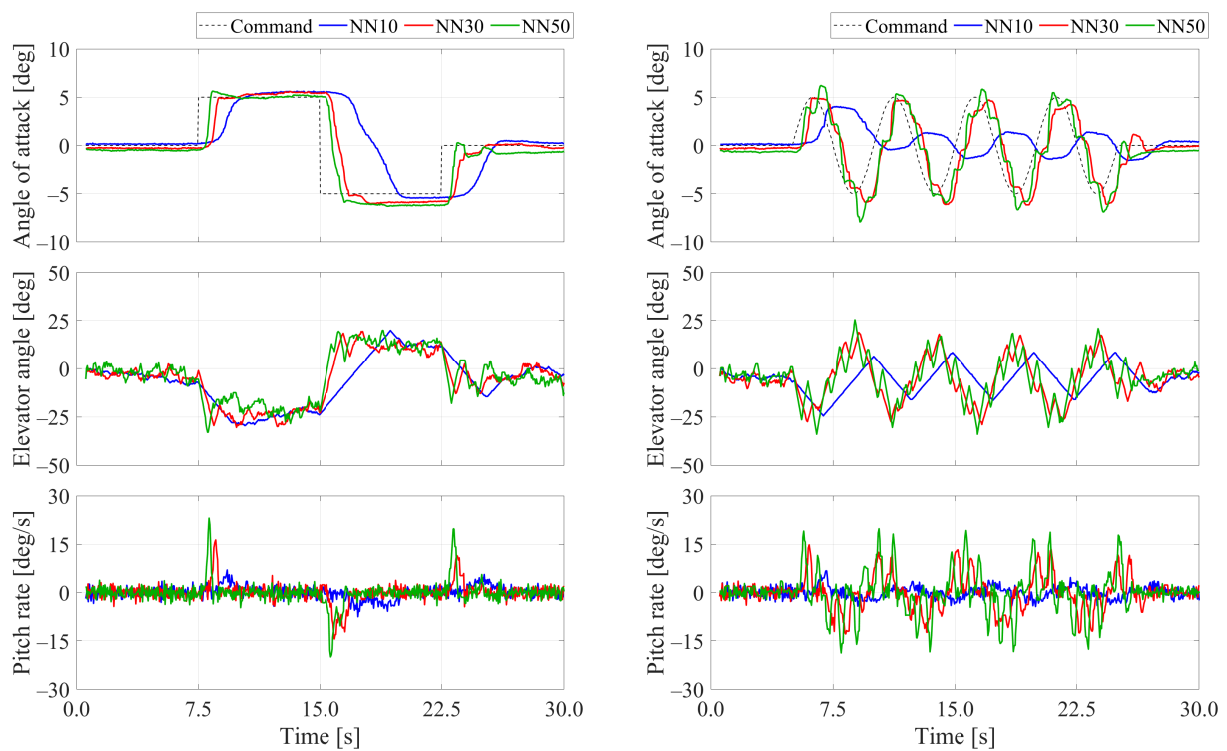


Figure 6. Training results. The solid lines are mean learning curves and the shaded areas are one standard deviation over 5 random seeds.

Figure 7 shows the time histories of the measured angle of attack, elevator angle, and pitch rate when the wind speed was 14 m/s. In the results with the doublet commands in Figure 7a, all neural network controllers successfully followed the angle-of-attack commands. The differences between the controllers were mostly evident in the transition phases. In the transition phase at 15 s, the elevator angle increased with the maximum action rates of the controllers. The pitch rate showed correlated peaks in the transition phases. The amplitude of the pitch rate increased, and the width decreased, for the controllers with higher action rates. Small angle-of-attack errors remained in the steady states. The maximum angle-of-attack errors in the -5° steady state were 0.055° , 0.52° , and 0.87° for NN10, NN30, and NN50, respectively. In principle, steady-state error could occur based on the balance between the angle-of-attack error and action penalty. A large action penalty could encourage the agents to remain in their states even with a small error rather than incur the action penalty. However, the accuracy of NN30 and NN50 could be as good as the accuracy of NN10 in theory. This indicated that learning parameters could be tuned further to increase accuracy. Figure 7b shows the time histories of the measured parameters with the sinusoidal commands. For NN10, even though the elevator moved at the maximum rate, it did not produce a large pitch rate to follow the command. The sinusoidal command was followed for NN30 and NN50. However, spiky fluctuations were observed in the time histories of the elevator angle and pitch rate. The fluctuations in the elevator angle indicated that the neural networks used the maximum amplitudes of the action rates and did not use other action choices. This behavior resulted in the fluctuations in the angle of attack.

Figure 8 shows the results obtained at wind speeds of 10 and 20 m/s. The results for NN10 with the doublet command are shown in Figure 8a. A higher wind speed led to more instantaneous responses in the transition phases because larger pitching moments were produced, as seen in Equation (1). The results for NN10 with the sinusoidal command are shown in Figure 8b. The variations in the angle of attack increased with the wind speed. Due to friction, a lower wind speed did not produce a pitching moment that was sufficiently large to vary the angle of attack. The results for NN50 with the sinusoidal command are shown in Figure 8c. The time histories of the elevator angle exhibited larger fluctuations for NN50 compared to NN10. Additionally, the fluctuations in the angle of attack increased with the wind speed. This was considered to be because of delay.

To summarize, the controllers with larger action rates showed better performance in terms of following the angle-of-attack commands. However, larger action rates and higher wind speeds caused fluctuations in the elevator angle and angle of attack, specifically in the cases with the sinusoidal commands.



(a) With doublet command.

(b) With sinusoidal command.

Figure 7. Pitch control results at wind speed of 14 m/s. NN10, NN30 and NN50 were used to follow doublet and sinusoidal commands.

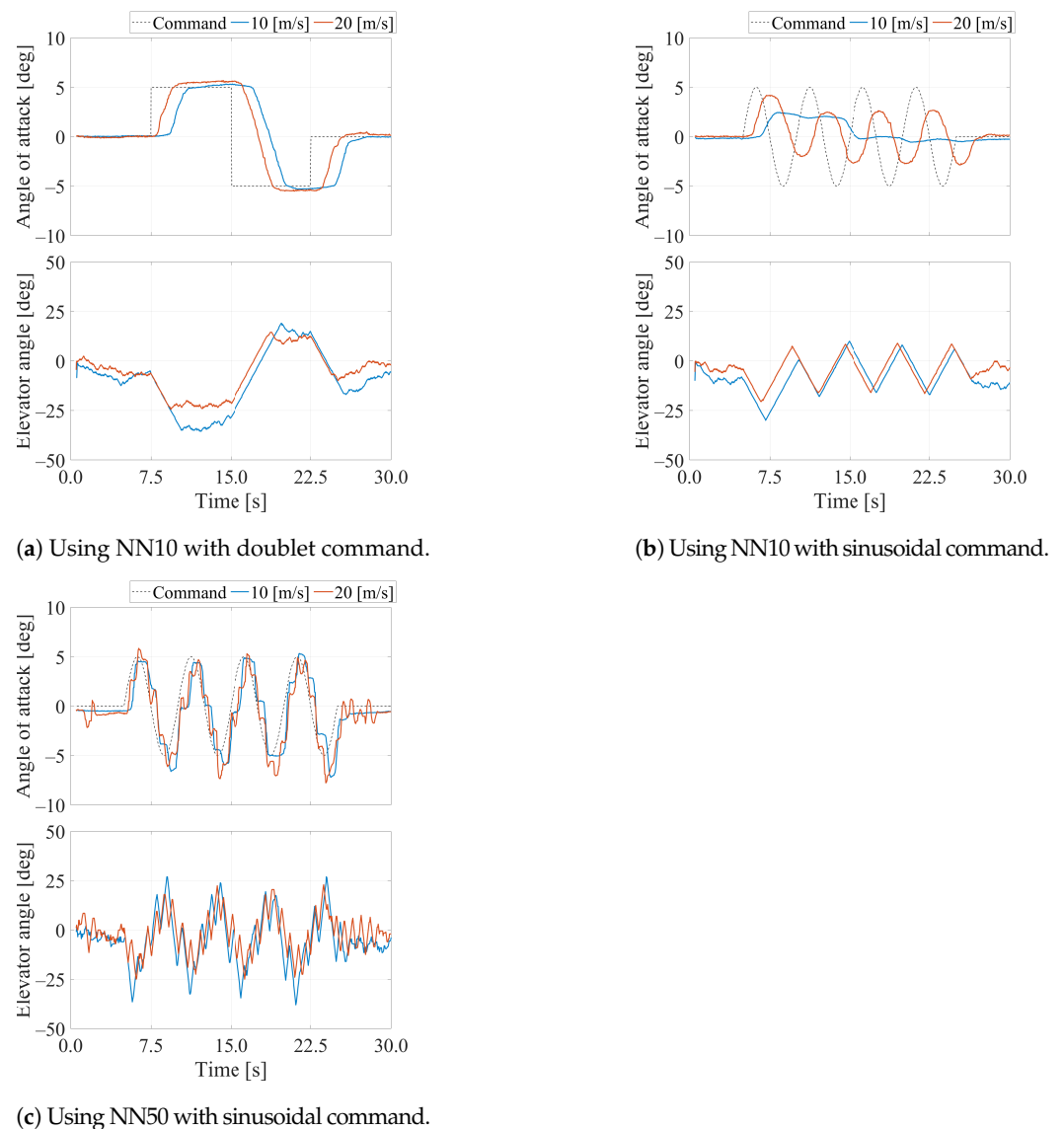


Figure 8. Pitch control results at wind speed of 10 and 20 m/s. NN10 was used to follow doublet (a) and sinusoidal (b) commands. NN50 was used to follow sinusoidal command (c).

4. Effects of Friction and Delay on Pitch Control

The effects of friction and delay on pitch control were investigated through simulation. Three theoretical models were tested. The first was the linear model that was used in the training. The second model included the friction in Equation (3) and the delay in Equation (12) with $t_0 = 15$ ms. The third model included friction and delay with $t_0 = 100$ ms. The second model considered only communication delay, and the third model included a larger offset delay, when compared to the second model, to examine the effect of additional delay. The longer time delay of $t_0 = 100$ ms was empirically selected. For the simulation with delay, the control outputs at each time step were queued in a buffer with individual delay times, which was assigned based on Equation (12). The command was activated to move the elevator after the delay time had passed.

Figure 9 shows the experimental and simulated time histories of the angle of attack with the doublet and sinusoidal commands for NN10 at a wind speed of 10 m/s. This condition was the least effective in producing pitching moment variations. Hence, it was considered that this condition demonstrated the effect of friction clearly. When the doublet command was applied, instantaneous changes were observed in the transition phases for the linear model. However, the starting times of the changes did not agree with the

experimental results. When friction was included in the model, the starting times of the instantaneous changes were delayed, and they agreed with the experimental results. When the sinusoidal command was applied, the effect of friction was evident over the entire time history of the angle of attack. A sinusoidal-like cyclic curve was obtained for the linear model, whereas flattened curves were obtained for the models that included friction. The flattened curves were closer to the experimental results. These observations indicated that friction affected the responsiveness of control. However, the effect of the size of the assumed time delay t_0 was negligible.

Figure 10a shows the experimental and simulated time histories of the angle of attack and elevator angle with the sinusoidal command for NN50 at a wind speed of 20 m/s. The magnified view of the time histories for the duration of 5–15 s is shown in Figure 10b. This condition was the most effective in producing pitching moment variations. The time histories of the elevator angle showed spiky fluctuations for the model that included delay. The fluctuations were larger and the time histories were more similar to the experimental results for the model with $t_0 = 100$ ms compared to the model with $t_0 = 15$. In principle, the fluctuations in the elevator maneuver are inefficient in terms of the action penalty in Equation (11). The result for the linear model did not show fluctuations. Hence, it was considered that delay induced the fluctuating behavior. In the time histories of the angle of attack, the results for the linear model and the model with friction and delay ($t_0 = 15$ ms) were smooth curves and did not show significant differences. The result for the model with friction and delay ($t_0 = 100$ ms) showed fluctuating behavior, and it was similar to the experimental result. The aforementioned results indicate that the effective delay in experiments should be assumed to be larger than communication delay. Considering that only communication delay can be directly measured, it would be difficult to predict the actual control performance without performing experiments.

For quantitative evaluation, the root mean square errors (RMSEs) for the angle of attack were calculated at different wind speeds. The results are shown in Figure 11. The circles represent the experimental results. Three tests per condition were conducted to verify repeatability. The experimental curves plotted in Figures 7–10 are the results with medium RMSE values among the three repetitive tests. When the doublet command was applied, the RMSE increased as the wind speed decreased. This was because the pitching moment produced by the elevator deflection decreased with the wind speed, and thus, the control was less effective. When the sinusoidal command was applied, the RMSE increased with the wind speed for NN10. This was because the variations in the angle of attack were larger at the higher wind speed, as shown in Figure 8b. The larger variations resulted in increased error compared to the flattened curve at the lower wind speed. For NN50, the RMSE increased slightly with the wind speed owing to the error caused by the fluctuations. The RMSE was the lowest for the linear model in the case of the doublet and sinusoidal commands because the same model was used in simulation as in training. The results obtained using the model with friction and delay ($t_0 = 100$ ms) agreed well with the experimental results. Specifically, the results for NN30 and NN50 at higher wind speeds were more accurately predicted because the model with the delay ($t_0 = 100$ ms) captured the fluctuating behavior.

The RMSE values averaged over the wind speeds and repetitive tests are listed in Table 2. It was essential to include friction to obtain accurate predictions. The model with delay ($t_0 = 100$ ms) provided accurate results, particularly for the sinusoidal command. Overall, the neural networks with larger action rates showed lower RMSEs. In the experiment, the RMSE values decreased from 3.42° for NN10 to 1.99° for NN50 with the doublet command. This indicated that the neural networks were successfully trained to utilize the merit of larger action rates. The larger action rates improved the control performance. However, they tended to cause fluctuating behavior owing to delay. This may result in unstable control in certain applications. In this regard, the delay in the system should be minimized and/or be considered in the training. In addition to delay, other aspects of the system such as aeroelasticity and aeroservoelasticity, which were not investigated

in this study, could affect the control performance and thus need to be considered in other applications.

In the previous studies of deep reinforcement learning for UAV's attitude control [19,20], convergence to target attitudes in simulation was a main basis for applicability evaluation. This study successfully highlighted the differences in control performances for simulations and experiments. The post-analysis presented the effects of the delay and friction, and their dependency on wind speeds and control commands. This highlights the difficulty is predicting the actual real world performance a priori based only on simulation results. Therefore, we suggest that experimental testing is essential to validate the performance for neural-network-based controllers trained through deep reinforcement learning.

One potential method for suppressing the fluctuating behavior, or mitigating the reality gap effect in general, is identifying a model with high fidelity and training a controller using the model. For example, in this study, delay was modeled by estimating the communication and mechanical delays. However, this might be challenging to estimate in other systems. In contrast to this case-specific approach, another approach is to train a controller with model uncertainties. In this approach, a few parameters in a model are set randomly at every episode such that the controller learns to be robust to uncertainties. This approach is referred to as domain randomization [27]. The use of these approaches to mitigate the reality gap will be investigated in the future.

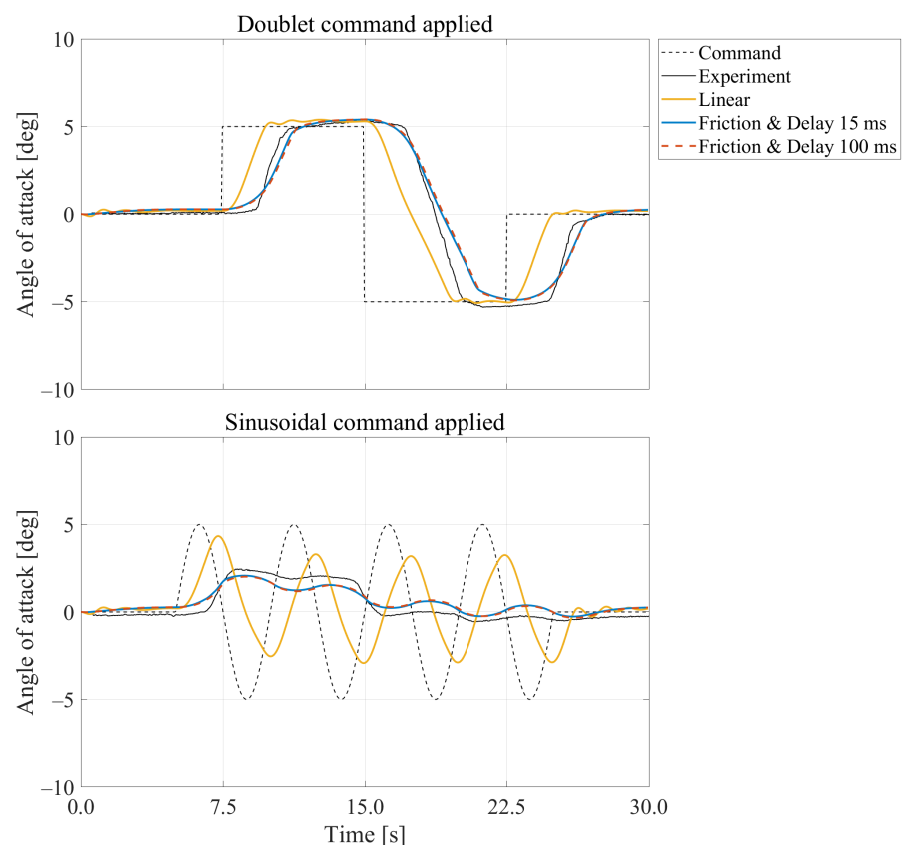


Figure 9. Simulated pitch control results for NN10 with doublet (**top**) and sinusoidal (**bottom**) commands at wind speed of 10 m/s.

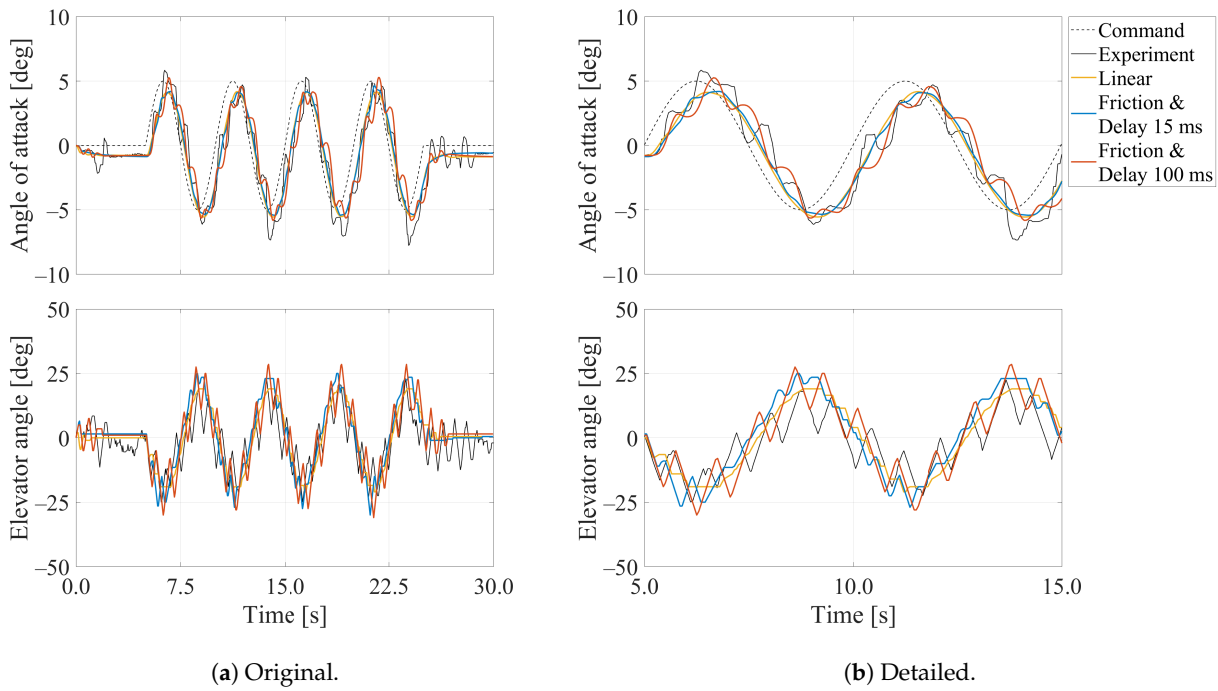


Figure 10. Simulated pitch control results for NN50 with sinusoidal command at wind speed of 20 m/s (a). Time range 5.0 s–15.0 s is magnified in (b).

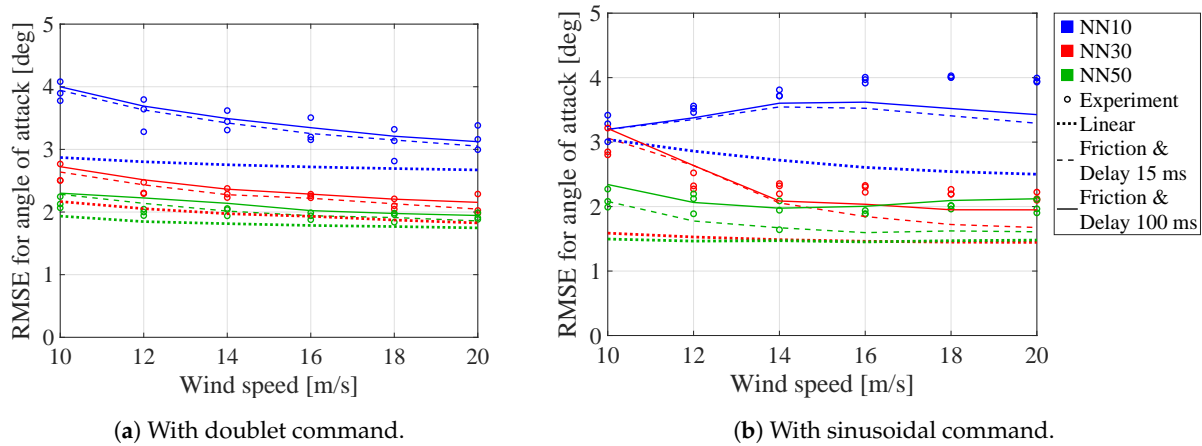


Figure 11. Root-Mean-Square Error (RMSE) distributions over wind speeds. RMSE was calculated for the angle of attack. Experimental results as well as theoretical simulation results using different models were plotted. NN10, NN30 and NN50 were used to follow doublet and sinusoidal commands.

Table 2. Mean RMSE over wind speeds for NN10, NN30, and NN50 with doublet and sinusoidal commands.

Method	Doublet			Sinusoidal		
	NN10	NN30	NN50	NN10	NN30	NN50
Experiment	3.42	2.29	1.99	3.74	2.38	1.99
Simulation with linear model	2.75	1.97	1.82	2.71	1.49	1.47
Simulation with friction and delay 15 ms	3.41	2.29	2.02	3.39	2.17	1.73
Simulation with friction and delay 100 ms	3.48	2.37	2.10	3.46	2.31	2.10

5. Conclusions

A reinforcement learning approach based on A3C was applied to train neural-network-based pitch controllers. Three controllers with different action choices, that is, elevator rates, were designed. The controllers were successfully trained, and their control performance was experimentally investigated through wind tunnel tests.

The experimental investigations demonstrated that the controllers with larger action rates showed better performance in terms of following angle-of-attack commands. The RMSE values decreased from 3.42° for NN10 to 1.99° for NN50 in the case of the doublet command. This indicated the controllers could be successfully trained to utilize the merit of larger action rates. The controller with a smaller action rate experienced the effect of friction, which degraded the responsiveness of control, specifically at low wind speeds.

In contrast, the larger action rates resulted in fluctuating behaviors in elevator maneuvers at high wind speeds. The numerical investigations indicated that the fluctuating behavior was caused by delay. The effective delay in the experiments was considered to be larger than the measured communication delay based on the comparison between the numerical and experimental results. This demonstrated that it would be difficult to predict the actual control performance without performing experiments.

As described in the introduction, most previous studies of neural-network-based reinforcement learning in the context of flight control have focused on theoretical control performance using simulation. The contribution of this study is to investigate the control performance of these types of machine learning based flight controllers in physical wind tunnel tests. The effects of friction and delay were discussed, which are common concerns in actual applications but are not always focused on in simulation. The method presented for designing controllers is considered to be applicable to different attitude control tasks, such as roll and yaw control. The understanding of the effect of the reality gap on pitch control can be generalized to the different control tasks. In the future, we will investigate approaches for mitigating the reality gap.

Author Contributions: Conceptualization, D.W., S.A.A.-E. and S.W.; methodology, D.W. and S.A.A.-E.; investigation, D.W. and S.A.A.-E.; data curation, D.W. and S.A.A.-E.; writing—original draft preparation, D.W.; writing—review and editing, D.W., S.A.A.-E. and S.W. All authors have read and agreed to the published version of the manuscript.

Funding: A part of this work was funded by JSPS KAKENHI (grant number JP19K04850). This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 679355).

Acknowledgments: The authors would like to thank Lee Winter from the University of Bristol Wind Tunnel Laboratory, for his invaluable support and work during the assembly of the experimental platform used to carry out the tests presented in this paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Luca, M.D.; Mintchev, S.; Heitz, G.; Noca, F.; Floreano, D. Bioinspired Morphing Wings for Extended Flight Envelope and Roll Control of Small Drones. *Interface Focus* **2017**, *7*, 1–11. [[CrossRef](#)]
2. Chang, E.; Matloff, L.Y.; Stowers, A.K.; Lentink, D. Soft Biohybrid Morphing Wings with Feathers Underactuated by Wrist and Finger Motion. *Sci. Robot.* **2020**, *5*, 1–14. [[CrossRef](#)] [[PubMed](#)]
3. Noll, T.E.; Ishmael, S.D.; Henwood, B.; Perez-Davis, M.E.; Tiffany, G.C.; Madura, J.; Gaier, M.; Brown, J.M.; Wierzbanski, T. *Technical Findings, Lessons Learned, and Recommendations Resulting from the Helios Prototype Vehicle Mishap*; NASA Technical Reports Server; NASA: Washington, DC, USA, 2007.
4. Rodriguez, D.L.; Aftosmis, M.J.; Nemecek, M.; Anderson, G.R. Optimization of Flexible Wings with Distributed Flaps at Off-Design Conditions. *J. Aircr.* **2016**, *53*, 1731–1745. [[CrossRef](#)]
5. Julian, K.D.; Kochenderfer, M.J. Deep Neural Network Compression for Aircraft Collision Avoidance Systems. *J. Guid. Control Dyn.* **2019**, *42*, 598–608. [[CrossRef](#)]
6. Gu, W.; Valavanis, K.P.; Rutherford, M.J.; Rizzo, A. A Survey of Artificial Neural Networks with Model-based Control Techniques for Flight Control of Unmanned Aerial Vehicles. In Proceedings of the 2019 International Conference on Unmanned Aircraft Systems (ICUAS), Atlanta, GA, USA, 11–14 June 2019; pp. 362–371.

7. Ferrari, S.; Stengel, R.F. Classical/Neural Synthesis of Nonlinear Control Systems. *J. Guid. Control Dyn.* **2002**, *25*, 442–448. [[CrossRef](#)]
8. Dadian, O.; Bhandari, S.; Raheja, A. A Recurrent Neural Network for Nonlinear Control of a Fixed-Wing UAV. In Proceedings of the 2016 American Control Conference (ACC), Boston, MA, USA, 6–8 July 2016; pp. 1341–1346.
9. Kim, B.S.; Calise, A.J.; Kam, M. Nonlinear Flight Control Using Neural Networks and Feedback Linearization. In Proceedings of the First IEEE Regional Conference on Aerospace Control Systems, Westlake Village, CA, USA, 25–27 May 1993; pp. 176–181.
10. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing Atari with Deep Reinforcement Learning. *arXiv* **2013**, arXiv:t1312.5602.
11. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-Level Control Through Deep Reinforcement Learning. *Nature* **2015**, *518*, 529–533. [[CrossRef](#)] [[PubMed](#)]
12. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous Control with Deep Reinforcement Learning. *arXiv* **2015**, arXiv:1509.02971.
13. Gu, S.; Lillicrap, T.; Sutskever, I.; Levine, S. Continuous Deep Q-Learning with Model-based Acceleration. *arXiv* **2016**, arXiv:1603.00748.
14. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.P.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous Methods for Deep Reinforcement Learning. *arXiv* **2016**, arXiv:1602.01783.
15. Schulman, J.; Levine, S.; Moritz, P.; Jordan, M.I.; Abbeel, P. Trust Region Policy Optimization. *arXiv* **2015**, arXiv:1502.05477.
16. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.06347.
17. Reddy, G.; Wong-Ng, J.; Celani, A.; Sejnowski, T.J.; Vergassola, M. Glider soaring via reinforcement learning in the field. *Nature* **2018**, *562*, 236–239. [[CrossRef](#)] [[PubMed](#)]
18. Koch, W.; Mancuso, R.; West, R.; Bestavros, A. Reinforcement Learning for UAV Attitude Control. *arXiv* **2018**, arXiv:1804.04154.
19. Clarke, S.G.; Hwang, I. Deep Reinforcement Learning Control for Aerobatic Maneuvering of Agile Fixed-Wing Aircraft. In Proceedings of the AIAA SciTech Forum, Orlando, FL, USA, 6–10 January 2020.
20. Bøhn, E.; Coates, E.M.; Moe, S.; Johansen, T.A. Deep Reinforcement Learning Attitude Control of Fixed-Wing UAVs Using Proximal Policy Optimization. In Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS), Atlanta, GA, USA, 11–14 June 2019; pp. 523–533.
21. Rabault, J.; Kuchta, M.; Jensen, A.; Réglade, U.; Cerardi, N. Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control. *J. Fluid Mech.* **2019**, *865*, 281–302. [[CrossRef](#)]
22. Tang, H.; Rabault, J.; Kuhnle, A.; Wang, Y.; Wang, T. Robust active flow control over a range of Reynolds numbers using an artificial neural network trained through deep reinforcement learning. *Phys. Fluids* **2020**, *32*, 053605. [[CrossRef](#)]
23. Makkar, C.; Dixon, W.E.; Sawyer, W.G.; Hu, G. A New Continuously Differentiable Friction Model for Control Systems Design. In Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Monterey, CA, USA, 24–28 July 2005; pp. 600–605.
24. Nguyen, D.; Widrow, B. Improving the Learning Speed of 2-layer Neural Networks by Choosing Initial Values of the Adaptive Weights. *Int. Joint Conf. Neural Netw.* **1990**, *3*, 21–26.
25. Uhlenbeck, G.E.; Ornstein, L.S. On the Theory of the Brownian Motion. *Phys. Rev.* **1930**, *36*, 823–841. [[CrossRef](#)]
26. Kingma, D.P.; Ba, J. ADAM: A Method for Stochastic Optimization. *arXiv* **2014**, arXiv:1412.6980.
27. Peng, X.B.; Andrychowicz, M.; Zaremba, W.; Abbeel, P. Sim-to-Real Transfer of Robotic Control with Dynamics Randomization. In Proceedings of the Proceedings—IEEE International Conference on Robotics and Automation, Brisbane, QLD, Australia, 21–25 May 2018; pp. 3803–3810. [[CrossRef](#)]