Reinforcement Learning to Control Lift Coefficient Using Distributed Sensors on a Wind Tunnel Model

A. Guerra-Langan*, S. Araujo-Estrada†, and S. Windsor‡
Department of Aerospace Engineering, University of Bristol, Bristol, United Kingdom

Arrays of sensors distributed on the wing of fixed-wing vehicles can provide information not directly available to conventional sensor suites. These arrays of sensors have the potential to improve flight control and overall flight performance of small fixed-wing uninhabited aerial vehicles (UAVs). This work investigated the feasibility of estimating and controlling aerodynamic coefficients using the experimental readings of distributed pressure and strain sensors across a wing. The study was performed on a one degree-of-freedom model about pitch of a fixed-wing platform instrumented with the distributed sensing system. A series of reinforcement learning (RL) agents were trained in simulation for lift coefficient control, then validated in wind tunnel experiments. The performance of RL-based controllers with different sets of inputs in the observation space were compared with each other and with that of a manually tuned PID controller. Results showed that hybrid RL agents that used both distributed sensing data and conventional sensors performed best across the different tests.

I. Introduction

Small uninhabited aerial vehicles (SUAVs) are light and agile, and as such, they are characterised by low Reynolds number and low flight speed [1]. These characteristics make them suitable for operations in the atmospheric boundary layer (ABL) [2–5], but they also make them particularly susceptible to the effect of unsteady aerodynamic forces. These are of special interest when trying to respond to gusts and turbulence, avoid obstacles or perform rapid manoeuvres [6]. A potential method for improving flight control in these types of conditions is by directly controlling the aerodynamic coefficients during flight. This approach could also provide a means of enlarging the safe operational envelope and flight efficiency of SUAVs. To achieve this we propose the use of two bio-inspired concepts: artificial neural networks following a reinforcement learning (RL) approach, and distributed sensing.

The control of lift coefficient at low Reynolds numbers has been developed in recent years with the aim of reducing the impact of gusts on SUAVs [6, 7]. These works rely on aerodynamic models and theories to get an estimation of lift force and lift coefficient, and their relationship with pitch rate to design lift coefficient controllers for gust-rejection. The simulation studies were performed with a flat plate, which gave promising results. However, when tested in open loop in a towing tank [8] with a known gust, the performance deteriorated. This was attributed to the model not capturing key flow features of the gust recovery region.

In nature, it is thought that the ability of flying animals to sense the distribution of air flow and related forces over their wings may contribute to their robust and efficient flight. Being able to sense these distributions over their flexible articulated wings offers the potential for animals to adjust their wing shape or motion to respond to disturbances, improve efficiency or enhance manoeuvrability [9–16]. Research suggests that the use of bio-inspired distributed sensor systems over the wing of small and micro UAVs could offer similar benefits for artificial flyers [17–20]. However, the integration of many parallel channels of airflow information into conventional flight control architectures is not straightforward. The arrays do not give direct measurements of the dynamic state like a conventional sensor. The exploitation of novel sensors on UAVs requires either the design of new controllers which can directly use the raw information coming from the sensor system (e.g. [21], [22]), or the design of new processing algorithms that translate the sensor signals into parameters that will feed conventional (e.g. [23, 24]) or alternative controllers (e.g. [21, 25]).

The information provided by distributed pressure and strain sensor arrays has been used to better understand the relationship between the dynamic states and loads of an SUAV and the airflow around it. For instance, two different UAV platforms were tested in the wind tunnel and free flight, one to measure the force (strain) and the other to measure the flow (pressure) [19]. Data showed a linear relation between α and both types of sensors up to stall conditions and an

*PhD Student, ana.guerra.langan@gmail.com
†Research Associate
‡Senior Lecturer in Aerodynamics

increase in the variance for higher α. In addition, the outputs of the sensors showed that both controlled and natural gusts can be detected and that properties which are not encoded by the inertial measurement unit (IMU) can be measured or calculated. Following this, a wing model was instrumented with both distributed pressure and strain sensors. The aerodynamic loads and states were estimated using artificial neural networks (ANNs) [20]. Measurements captured detachment of the flow and non-linear behaviours such as hysteresis and rate dependent effects. This sensing array was then used for angle of attack control of a 1 degree-of-freedom (DOF) platform in wind tunnel experiments [21], and for airspeed control of a 3 DOF simulation model [22] by means of ANN-based controllers. The prior use of distributed pressure and strain sensing arrays for aerodynamic loads estimation and application on end-to-end control sheds light on the potential use of this sensing array to control aerodynamic coefficients during flight.

The work in [22] saw some of the benefits and limitations of using a supervised learning approach. Within machine learning, supervised learning is a subcategory defined by its use of labelled datasets to train algorithms to classify or predict outcomes accurately. The results in [22] saw that ANNs were capable of both dealing with the size and high-complexity of the distributed sensing system and of adapting to different situations not included in their training set, especially when using the full sensor suite. One of the limitations of this approach was that the controllers aimed to match the performance of the training set instead of learning to optimise or minimise the error between the desired and the actual parameter. ANNs have been used in aviation for limited applications which include replacing components of conventional flight control systems [26] such as gains [27], or as look-up tables for aircraft collision avoidance algorithms [28]. Limited research has been conducted on the use and safe application of machine learning for flight control, but some studies have used ANNs in conjunction with adaptive control [29], for gust detection and load mitigation on wind turbines [30], for stabilisation and trajectory control of a hexacopter [31], a flapping wing [32], or a model-free simulation of a UAV [33], and to attain robust non-linear control for active guidance of a finless rocket [34]. Most of these studies trained the ANNs following a supervised learning approach with their performance tested in simulation, rather than with physical platforms.

One of the alternative approaches to supervised learning is reinforcement learning (RL), which is a computational approach to learning from interaction, where trial-and-error search and delayed reward take an important role [35]. In this case, the neural networks are trained based on the experience of an agent interacting with the environment. RL algorithms model and learn complex nonlinear relationships between variables, deal with large observation spaces, learn from the environment to achieve specific goals, and optimise and generalise the solution with a trade-off between exploration and exploitation [35]. This approach has mostly been used in simulated environments and video-games, and less so on real-world platforms. However, RL has been used in SUAV flight control for a series of applications. An autonomous helicopter learnt to follow a series of manoeuvres using the Pegasus RL algorithm [36], with the pre-built learning framework being able to adapt to changes in the dynamics of the vehicle. A PID controller for multirotor attitude control was compared to three different RL agents in [37]. In this work, Deep Deterministic Gradient Policy (DDGP), Trust Region Policy Optimisation (TRPO) and Proximal Policy Optimisation (PPO), were trained and compared in simulation and the PPO algorithm showed superior performance overall, including outperforming a conventional controller. Fixed-wing UAVs have also seen RL algorithms used for control, showing high levels of performance for attitude control in simulation [38], for angle of attack control in wind tunnel tests [25], or for perched landing [39–41]. A Q-learning RL algorithm was used to attain a policy based agent to find the best flight profile for perching manoeuvre in open loop flight tests [39]. Flight tests of the same platform in closed-loop configuration [40] followed on from this, where a Deep Q-Network (DQN) algorithm controlled the aircraft during flight but suffered from the reality gap between the simulation environment it had learnt from and flight tests. More recently, flight tests showed a reduced reality gap when an improved version of this controller was developed using PPO [41]. The PPO algorithm [42] has become the OpenAI's algorithm of choice and one of the most commonly used due to its ease of use and good performance. This algorithm provides the data efficiency and reliable performance features typical of TRPO whilst being simpler and easier to implement [43].

The aim of the work presented here was to estimate aerodynamic coefficients by means of distributed pressure and strain sensors over the wing of a fixed-wing SUAV and to directly control these using an RL based approach. An ANN aerodynamic coefficients estimator is presented, which was trained to estimate lift and drag coefficient using pressure and strain sensor readings together with pitch rate. These coefficients were then used to train a series of RL agents to control a 1 DOF platform in simulation to track the estimated lift coefficient ($\widehat{C_L}$). The agents were trained using the PPO algorithm with different observation and action spaces, and the best performing controllers then tested in the wind tunnel (WT) to validate their simulation performance and to assess their applicability on a physical platform. The $\widehat{C_L}$ root-mean-square-error (RMSE) was calculated and used to compare the performance of the different agents for this task. A PID controller was also manually tuned in the wind tunnel for $\widehat{C_L}$ control with the aim of comparing the behaviour of linear and nonlinear controllers for this task. If applied on an SUAV, the controllers proposed in this work could potentially increase the manoeuvrability of the vehicle, enlarge the safe operational envelope and help with gust alleviation.

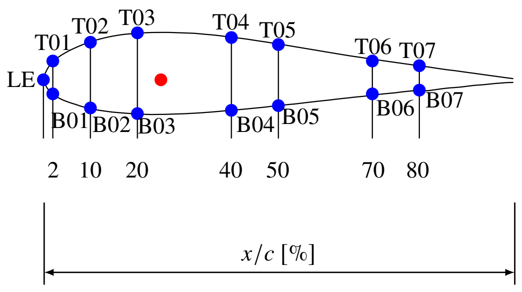
The methods used in this work are described in detail in Section II. The performance of the aerodynamic coefficients estimator is presented and discussed in Section III, and the performance of the $\widehat{C_L}$ controllers in wind tunnel experiments is shown in Section IV. The benefits and limitations of the different controllers are discussed in Section V, together with the contributions to the field and how other work could build on the methods and results presented in this paper. Finally, conclusions are drawn in Section VI.

II. Methods

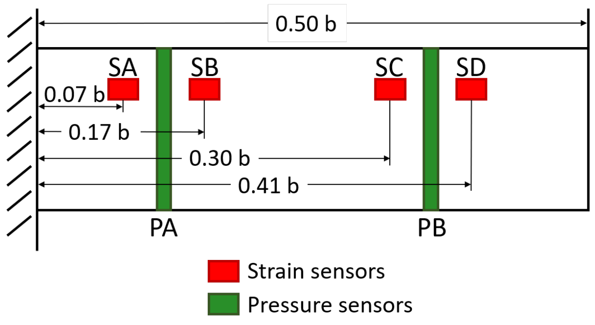
The first stage of this study was to wind tunnel test an instrumented 1 DOF experimental platform to record distributed sensor readings and aerodynamic loads over a range of conditions. This dataset was then used to build a lift and drag coefficient (C_L, C_D) estimator, and was integrated into an empirical flight dynamics model with the aim of replicating the behaviour of the physical platform in simulation. The simulation environment was used to train a series of reinforcement learning agents to control lift coefficient. These agents were tested on the physical platform in wind tunnel experiments to assess their performance. This section describes the approach in more detail.

A. Wind Tunnel Dataset

The dataset used to build the empirical flight dynamics model for simulation was gathered from wind tunnel tests. This was done through a series of experiments which were carried out in the University of Bristols 2.13 m × 1.52 m (7 ft × 5 ft) low speed wind tunnel. These tests were performed using a semi-span wing of a WOT 4 Foam-E Mk2+ (Ripmax, Enfield, UK) radio control aircraft, instrumented with an array of 30 pressure and 4 strain sensors as per Fig. 1, which was the same wing model used in [20–22]. The wing was mounted on a ply and balsa wood half-fuselage built following the original outline design of the WOT 4 fuselage, with a larger tailplane for increased pitch control authority. The model was equipped with an inertial measurement unit (IMU) (Pixhawk 1, 3DR) and a servo motor (Hitec HS-5645MG) linked to the elevator. These together with an airspeed sensor on the wing were connected to a micro-controller unit (PJRC, Teensy 3.6). The model was mounted to the wind tunnel wall, supported by a shaft which allowed free motion around the pitch axis, or that could be driven by a large servo motor (Schneider Electric, LXM32MD30M2 and BMH1401P01F2A). The aerodynamic loads were measured using a load cell (ATI Industrial Automation, Miny 45) mounted between the wing support and the rig's shaft. The maximum angle of attack (α) value available given the size of the wind tunnel, the size of the fuselage and the position of the rig was approximately 29°. The maximum and minimum effective elevator deflection (δ_e) available was ±40°.



(a)



(b)

Fig. 1 Distributed sensing array: *a)* chord-wise (c) pressure array distribution taken from [20] and *b)* span-wise (b) strain array distribution

Two sets of experiments were carried out in the wind tunnel where the sensor readings, the dynamic states and the aerodynamic loads were recorded. First, servo driven tests where the aircraft model was moved to specific angles of attack from −20° to 29°, repeated four times at airspeeds $V = [8, 10, 12, 14, 16, 18, 20]$ m/s and pitch rates $q = [1, 5, 10, 20, 30, 40, 50]$ °/s. The angle of attack in these tests was measured by the encoder in the large servo motor.

The data gathered from these tests was used to build the empirical model and the aerodynamic coefficient estimator as described in Section II.B. Second, elevator frequency sweeps were undertaken with the aircraft pitch being driven by changes in the elevator deflection. The frequency sweeps ranged from 0.1 Hz to 10.0 Hz and were performed at each of $V = [8, 10, 12, 14, 16, 18, 20]$ m/s. Note that the angle of attack in these tests was given by the pitch angle measured by the IMU. The data gathered from these tests was used to derive a model of the pitching moment of the aircraft by means of the Output Error Method [44]. The model and the corresponding parameters are given in Appendix A.

B. Simulation framework: Empirical model

A 1 DOF empirical model of the experimental platform was built in Python 3.6. The mathematical model for the pitching moment of the experimental platform was derived from the frequency sweeps described above, and given by Eqs. A.1 and A.2, and the coefficients in Table A.2 in the Appendix. Figure 2 shows a block diagram of the 1 DOF empirical model used in simulation. The inputs to the controllers varied depending on the controller chosen in each test.

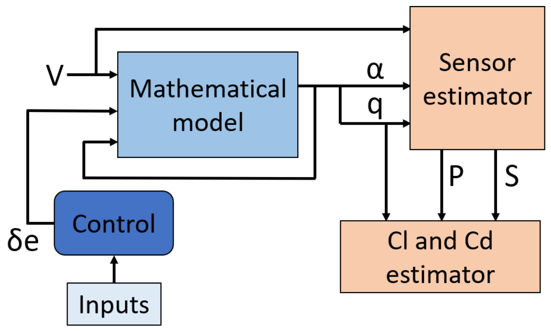


Fig. 2 Block diagram of the 1 DOF empirical model, with *P* representing the pressure sensor signals and *S* the strain sensor signals

The dataset gathered in Section II.A was included in the simulation framework by means of ANNs. These were used to serve as sensor estimators in the simulation framework, and as C_L and C_D estimators in both simulation and WT tests. All the ANNs were trained with the MLPRegressor function in the "Scikit-Learn v0.22.2.post1" package for Python [45], using the *tanh* activation function for 1000 epochs with a maximum number of validation failures set to 50. The inputs to the networks were normalised in [-1, 1]. The training and validation set used approximately 85% of the wind tunnel dataset, while the remaining 15% was used to test the performance of the ANNs. This evaluation was done with the "score" method of MLPRegressor, which returned the coefficient of determination (R^2) of the prediction.

a) **Pressure and Strain estimation**
The relationship between the state parameters and the sensor outputs was obtained by curve fitting the data with ANNs. A neural network was defined for each span-wise section of the wing, i.e., two for the 2 chord-wise pressure sensor arrays and four for the 4 strain sensors used, giving a total of 6 ANN sensor estimators. The details of the ANNs used in the estimators are given in Table 1.

b) **Aerodynamic coefficients estimation**
In this work the aerodynamic coefficients were estimated ($\widehat{C_L}$ and $\widehat{C_D}$) using the pressure and strain sensor information as inputs together with q as was suggested in [20]. ANNs were used due to the size of the dataset and its complexity. The lift and drag coefficients in the training set were calculated from the lift and drag forces measured by the loadcell during the wind tunnel tests (Section II.A). Both C_L and C_D were estimated from a single ANN (Table 1).
The servo-driven tests were carried out with all the control surfaces set to 0°. This meant that the distributed pressure and strain sensor information was aligned with the aerodynamic loads for these specific conditions. An assumption was made in the simulation model that the effect of the elevator on C_L and C_D for this platform was negligible.

The R^2 values for the testing set presented in Table 1 were all close to 1.0 indicating the ANNs could fit the data well. The sensor estimators were used to simulate the sensor readings over the wing of the vehicle at any given state,

Table 1 Summary of ANN estimators (pressure, strain and aerodynamic coefficients)

Estimated parameters	Inputs	Neurons in hidden layers	Number of outputs	Testing R^2
Pressure A (PA)	States $[\alpha, V, q]$	16-16	15	0.989
Pressure B (PB)				0.991
Strain A (SA)		8-16-8	1	0.996
Strain B (SB)				0.995
Strain C (SC)				0.993
Strain D (SD)				0.976
$\widehat{C}_L, \widehat{C}_D$	[q, PA, PB, SA, SB, SC, SD]	16-32-16	2	0.986

to replicate the information available on the physical platform. The aerodynamic coefficient estimator was designed to estimate these parameters given the sensor readings, with the aim of being applicable in both simulation and wind tunnel testing or future outdoor flight experiments. Figure 3 shows that the estimated C_L and C_D correlates well with the measured values, calculated from the loadcell data. The errors in the estimation are presented in Table 2, where the RMSE value and the RMSE value relative to the measurement range (%MR) is shown in brackets for the testing dataset. The measurement range was obtained from the full original dataset, by getting the maximum and minimum measured values. The data presented in Table 2 indicates that overall, the aerodynamic coefficients can be estimated with an error of 0.73%MR for C_L and 2.78%MR for C_D, showing a similar performance at high and low angles of attack.

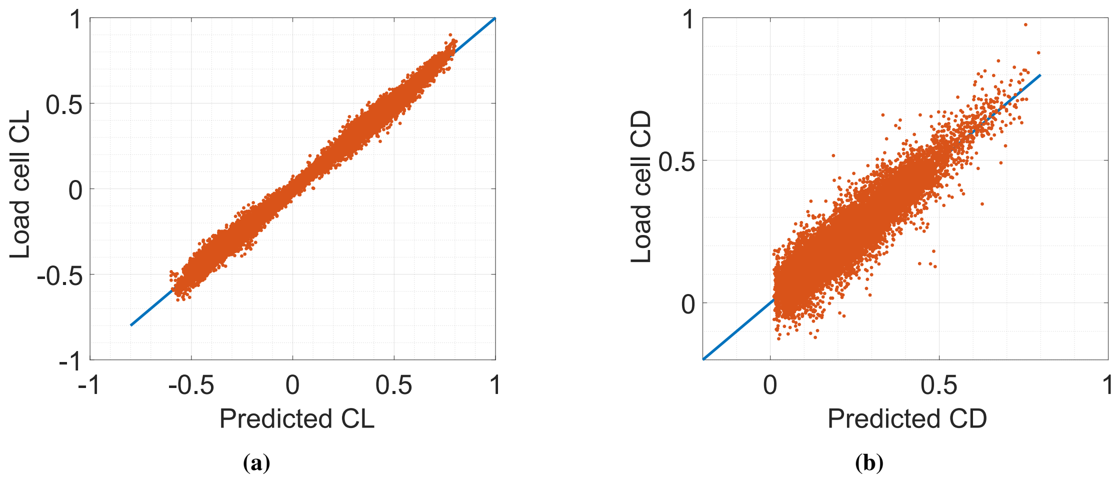


Fig. 3 Real against predicted *a)* C_L, and *b)* C_D, for the testing set

Table 2 RMSE and %MR values for estimated C_L and C_D from the testing dataset

Variable	Measurement range	Overall RSME (%MR)	RSME (%MR) $\alpha > 11°$
C_L	3.29	0.024 (0.73)	0.025 (0.76)
C_D	1.26	0.035 (2.78)	0.035 (2.78)

C. Reinforcement Learning Controller

The work presented aimed to control the lift coefficient using a policy-based reinforcement learning approach. Generally, RL methods depend on the interaction between an agent capable of taking decisions and its environment. In this way, at time t, the agent receives a state or observation s_t after interacting with the environment, and uses this information to compute the action it will perform next, a_t. The learning occurs with the help of the reward function, which gives a measure of how good the action was at that time, $r_t = R(s_t, a_t)$, while it transitions into the next state, s_{t+1}. The end goal is for the agent to maximise its return, $R(\tau)$ which is the cumulative reward over a trajectory. This trajectory is defined as the sequence of states s_i, actions a_i and rewards r_i taken ($\tau = s_0, a_0, r_1, s_1, a_1, r_2, ...$).

The proximal policy optimisation (PPO) was the RL algorithm of choice in this study. This is a policy-gradient method proposed by Schulman *et al.* [42] which optimises a surrogate objective function using a stochastic gradient ascent. Stochastic policy gradient methods work by constructing an objective function and iteratively estimating its derivative with respect to policy parameters by taking small steps. PPO comes as an improvement to the trust region policy optimisation (TRPO), by modifying the surrogate objective function and constraining the size of the update steps with a clipped objective function.

The surrogate objective proposed by PPO is:

$$J^{CLIP}(\zeta) = \widehat{\mathbb{E}}_t \left[min(v_t(\zeta)\widehat{A}_t, clip(v_t(\zeta), 1-\epsilon, 1+\epsilon)\widehat{A}_t) \right] \tag{1}$$

$$v_t(\zeta) = \frac{\pi_\zeta(a_t|s_t)}{\pi_{\zeta_{old}}(a_t|s_t)} \tag{2}$$

where $\widehat{\mathbb{E}}_t[...]$ is the expectation, $\widehat{A}_t$ is an estimator of the advantage function at timestep t, ϵ is a hyperparameter, $v(\zeta)$ is the probability ratio between the updated policy and the current policy, and π_ζ is a stochastic policy parameterised by ζ. The $clip$ function clips the ratio to be no more than $1+\epsilon$ and no less than $1-\epsilon$. This function discourages too large updates, limiting the difference between π_ζ and $\pi_{\zeta_{old}}$ by including the probability ratio only when it makes the objective worse.

The PPO2 algorithm provided by the “Stable Baselines 2.10.1” [43] package in Python was used to train the agents presented in this work. This method combines the PPO algorithm described above with an actor-critic structure, which at the same time combines value functions with explicit representation of the policy. This structure uses two separate neural networks, one (actor) to select actions, and the other (critic) to evaluate them. The objective function in this case needs to account for the learnt state-value function $\mathbb{V}(s)$ and an entropy term $\mathbb{S}$ to encourage sufficient exploration,

$$J_t^{CLIP+VF+\mathbb{S}}(\zeta) = \widehat{\mathbb{E}}_t \left[J_t^{CLIP}(\zeta) - c_1 J_t^{VF}(\zeta) + c_2 \mathbb{S}[\pi_\zeta](s_t) \right] \tag{3}$$

where c_1 and c_2 are coefficients, $\mathbb{S}$ denotes an entropy bonus, and J_t^{VF} is a squared-error loss $(\mathbb{V}_\zeta(s_t) - \mathbb{V}_t^{targ})^2$.

Overall, this algorithm attains the data efficiency and reliable performance features typical of TRPO whilst having multiple agents, and being simpler and easier to implement.

D. Training approach

The PPO2 algorithm described above was used to train a series of agents in simulation to control the 1 DOF platform with the aim of tracking a desired lift coefficient.

Fourteen different specifications were trialled during the training of the agents (i.e., 7 observation spaces × 2 actions spaces × 1 reward function). Each of these specifications were trained five times, with different seeds to ensure generalisation. The neural networks were trained using the default settings of “MLPPolicy” provided by Stable-baselines, which uses a fully-connected multi-layer perceptron (MLP) with two hidden layers of 64 neurons each with *tanh* activation functions. The inputs to the controllers were normalised in [-1 1], and the output of all agents was the elevator angle rate, which was chosen so that the full range of elevator deflection angles could be covered by a discrete action space. Table 3 shows a summary of this, where PS was the full pressure and strain sensor suite, $States = [\alpha, V, q, \delta_e]$, r was the reward function, C_{L_d} was the desired C_L, and $\dot{\delta}_{e_{max}}$ was the maximum elevator deflection rate set to 120 °/s. The reward function was empirically designed with the aim of minimising the error between the desired lift coefficient and the estimate by producing negative rewards proportional to the absolute value of the error.

The pitching moment model derived from wind tunnel tests and used in the simulation framework depended on airspeed as shown in Table A.2. Because of this, it was important that all agents included V directly or indirectly as an input in their observation space as well as enough information to derive the aerodynamic coefficients. The full sensor suite alone contained enough information about V, used for airspeed control in [22] and to estimate the aerodynamic loads [20]. Agent 1 was trained using sensor information exclusively; agents 2-3 were trained with the full states of the platform or a subset of the state; and agents 4-7 were hybrid versions that used different combinations of sensor

Table 3 Summary of trained agents

Agent ID	Input layer / Observation space	Hidden layer	Action space, °/s	Reward function
1	PS	64-64	Discrete: $[\pm 120, \pm 50, \pm 5, 0]$ Continuous: $[-1, 1] \times \dot{\delta}_{e_{max}}$	$r = -\lvert C_{L_d} - \widehat{C}_L \rvert$
2	States			
3	$[\alpha, V, q]$			
4	PS + States			
5	$\widehat{C}_L + V$			
6	$\widehat{C}_L + PS$			
7	$PS + q$			

information and states. In particular, agents 6-7 both contained the same information in different forms: PS and q, since the aerodynamic coefficients were estimated using these parameters.

During the training process, each episode had a duration of 30 s with 600 timesteps, assuming a control rate of 20 Hz. The wind speed was defined as a constant in each episode, randomly selected as an integer within the range of 8 to 20 m/s. The initial α was within the range of -30 to 30°, the initial q was within the range of -5 to 5 °/s and the initial δ_e was within the range of -40 to 40°. C_{L_d} was defined by a "doublet" signal, shown in Fig. 4. In this case, the base C_{L_d} value was equal to the initial $\widehat{C}_L$, the ramp time t_c was 0.5 s and the step amplitude (A) was randomly chosen as a value within the range of 0.1 to $min(|C_{L_{min}}(V) - \widehat{C}_{L_{init}}|, |C_{L_{max}}(V) - \widehat{C}_{L_{init}}|)$. Here, $C_{L_{min}}(V)$ and $C_{L_{max}}(V)$ were the minimum and maximum C_L values for $q = 1$ °/s at the specified V given by the dataset used to build the empirical model. In this way, the desired C_L was never below or over the minimum and maximum possible values at the given airspeed, respectively. In the doublet wave, C_L was held at $\pm A$ for a duration of 10 s.

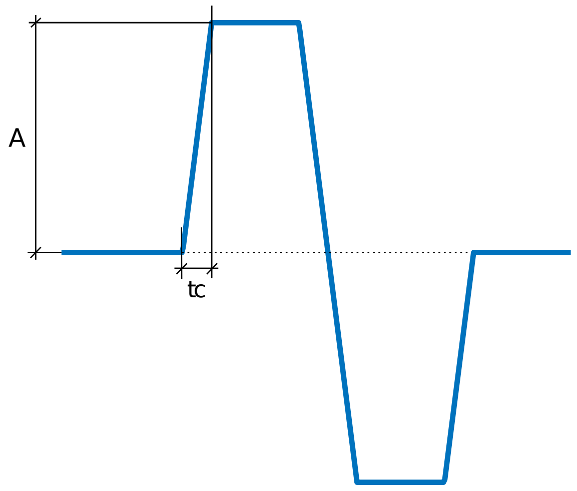


Fig. 4 Doublet signal

The training process saw each agent trained five different times with different seeds in order to reduce the probability of getting high or low training scores by chance. During the training, the algorithm checked the mean return of the previous one hundred episodes every 1000 episodes, and the best agent based on this value was saved. The *final* agent used per batch of five, was chosen based on the mean return obtained after running the simulation for 50 random test cases. This gave a total of fourteen agents per control task, which was downsized to seven by selecting the best performance based on observation space, not considering whether it used a discrete or continuous action space (Table B in the Appendix). The reward during training is shown in Figure B.1 in the Appendix for the selected agents, and their behaviour in simulation is described in Section IV.

E. Experimental Approach

The controllers presented in this work were tested and validated in the University of Bristol's 2.13 m × 1.52 m (7 ft × 5 ft) low speed wind tunnel following the setup in Fig. 5. Here, the aerodynamic coefficients estimator was used to get live estimations of C_L given the readings of the distributed sensing system. The inputs to the controller varied depending on the controller chosen in each test.

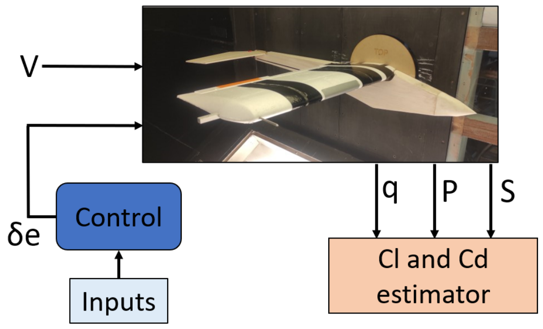


Fig. 5 Block diagram of the experimental 1 DOF platform in the wind tunnel setup

Seven agents, one per observation space, and a manually tuned PID $\widehat{C}_L$-controller (gains in Table A.3) were used to control the 1 DOF platform to track a C_L doublet signal (Fig. 4). The base C_L value was set to 0.2, the amplitude was defined as ±0.5 and the duration at each section was of 10 s with $t_c = 0.5$ s. The total duration of each experiment was of 42 s, and they were run at three different airspeeds, $V = 10, 15, 18$ m/s, which were set to a constant value during each test. The individual experiments were repeated five times in order to verify repeatability. The linear controller was used as a baseline to which to compare the RL-based controllers. The aim of these experiments was to study the application of the controllers on a physical platform for C_L control in the linear region.

Following on the results of these tests, two of the best performing RL agents and the PID controller were tested at a higher C_L value. This C_{L_d} was defined as the maximum C_L at each tested V, which was taken from the wind tunnel dataset in Section II.A (gathered in Table A.4). The tests were performed under two different conditions to study their robustness and analyse their performance and limitations:

1) Baseline: The tests consisted on setting $\delta_e = 0°$ for 5 s and then turning the controllers on for 35 s for a total duration of 40 s per test.
2) Perturbation 1: As above, but in this case the aileron and flaps were deflected to 40° for 0.5 s after 5 s of control; and for 10 s after 11 s.

These additional tests were aimed at investigating the performance of the controllers in the nonlinear region to set a basis for future work.

III. Performance of the aerodynamic coefficients estimator

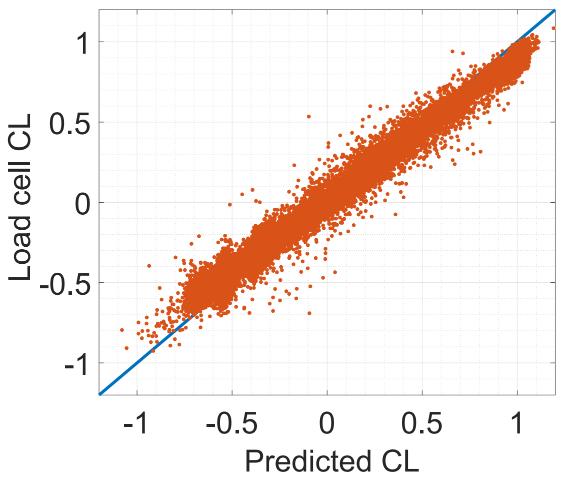
In this work, an ANN was used to estimate C_L and C_D given the readings of distributed sensor data together with pitch rate, as described in Section II.B. The estimator was trained with servo-driven data, with all control surfaces set to 0°. The effect of the elevator on $\widehat{C}_L$ and $\widehat{C}_D$ was assumed to be negligible.

Figure 6 shows the estimated aerodynamic coefficients against the real values, calculated from the load cell data.

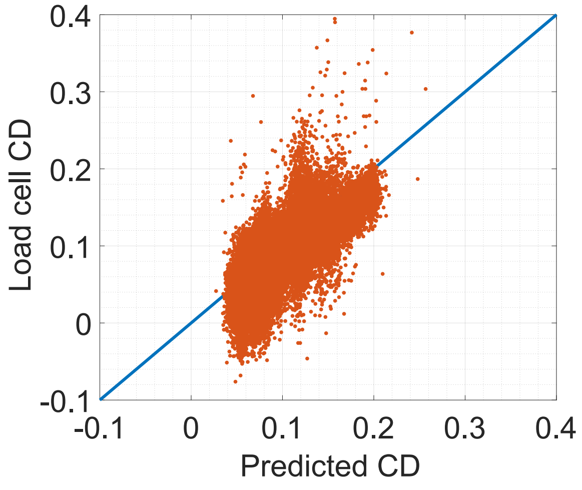
The performance of the aerodynamic coefficients estimator was assessed by calculating the RMSE value between the estimation and the calculated value from load cell readings. The computed RMSE values are given in Table 4 and the RMSE relative to the measurement range (%MR) is shown in brackets. The measurement range was taken from the original training set, as per Section II.B.

Table 4 RMSE and %MR values for estimated C_L and C_D in WT tests

Variable	Measurement range	Overall RMSE (%MR)	RMSE (%MR) $\alpha > 11°$
C_L	3.29	0.063 (1.92)	0.109 (3.31)
C_D	1.26	0.028 (2.22)	0.030 (2.34)



(a)



(b)

Fig. 6 Comparison between estimated aerodynamic coefficients and calculated values from load cell data: *(a)* C_L and *(b)* C_D

The RMSE values presented in Table 4 together with the data in Fig. 6 suggest that the estimators could be affected by random errors, such as differences in barometric pressure between the servo-driven WT tests performed to attain the training dataset and the validation tests presented in this work, or minor differences in the setup between the two sets of WT tests (i.e., rig or platform configuration). In addition, systematic errors were also present, resulting in over-estimates of the parameters due to extrapolation, which occurs when the estimator encounters states that were not given in the training set. The results in Table 4 suggest that the aerodynamic coefficients can be estimated with an overall RMSE of up to 2.22% of the measurement range. The data in Table 4 also shows the RMSE value specific to high angles of attack, $\alpha > 11°$, presenting an error of 3.31%MR for C_L and 2.34%MR for C_D. This is of special interest for C_L control around stall.

The %MR errors seen for C_L estimations in the RL wind tunnel tests were greater than those seen in Section II.B for the testing set (0.73%MR), and than those seen in [20] for lift load estimation (0.77%MR). The testing sets used to attain these values in Section II.B and [20] were gathered in the same set of wind tunnel tests as the training dataset, and following the same approach. This means that even though the datasets used for training and testing were different, the rig configuration and set-up remained the same and the testing dataset was uniformly distributed across the full range of α. The work presented in this paper was collected in a different set of tests where differences in rig and set-up could have had an effect on the results, and where the data collected was dependent on the control tests performed.

Further improvements in the estimation of the aerodynamic coefficients could be achieved by studying a greater variety of ANN architectures, by investigating the use of additional information as inputs to the estimator, studying the sensor layout following the approach in [22] for V control, and by increasing the training dataset to include perturbed state of the flow.

IV. Performance of the controllers

The seven RL agents presented in Section II.D were tested in wind tunnel experiments together with a manually tuned PID $\widehat{C}_L$-controller following the approach in Section II.E. In the results displayed in this section, the agents were clustered in different colours based on the inputs to their observation space. Agent 1 with "*PS* only" was purple, agents 2-3 with "states only" were in red tones, 4-5 hybrid versions with different state combinations in blue tones, and 6-7 hybrid versions with $PS + q$ given in different forms in green tones.

The time-histories for the repeat with median RMSE value among the five repetitive tests for each case in the wind tunnel tests are presented in Fig. 7a for $V = 15$ m/s. The same tests in simulation are presented in parallel in Fig. 7b for comparison. At the start, the elevator was set to 0° for all tests, and the controllers were switched on at $t = 2.5$ s. In addition, a summary of the overall performance of all the controllers is presented in Fig. 8, by the mean $\widehat{C}_L$ RMSE value across repeats at different airspeeds for WT and simulation tests.

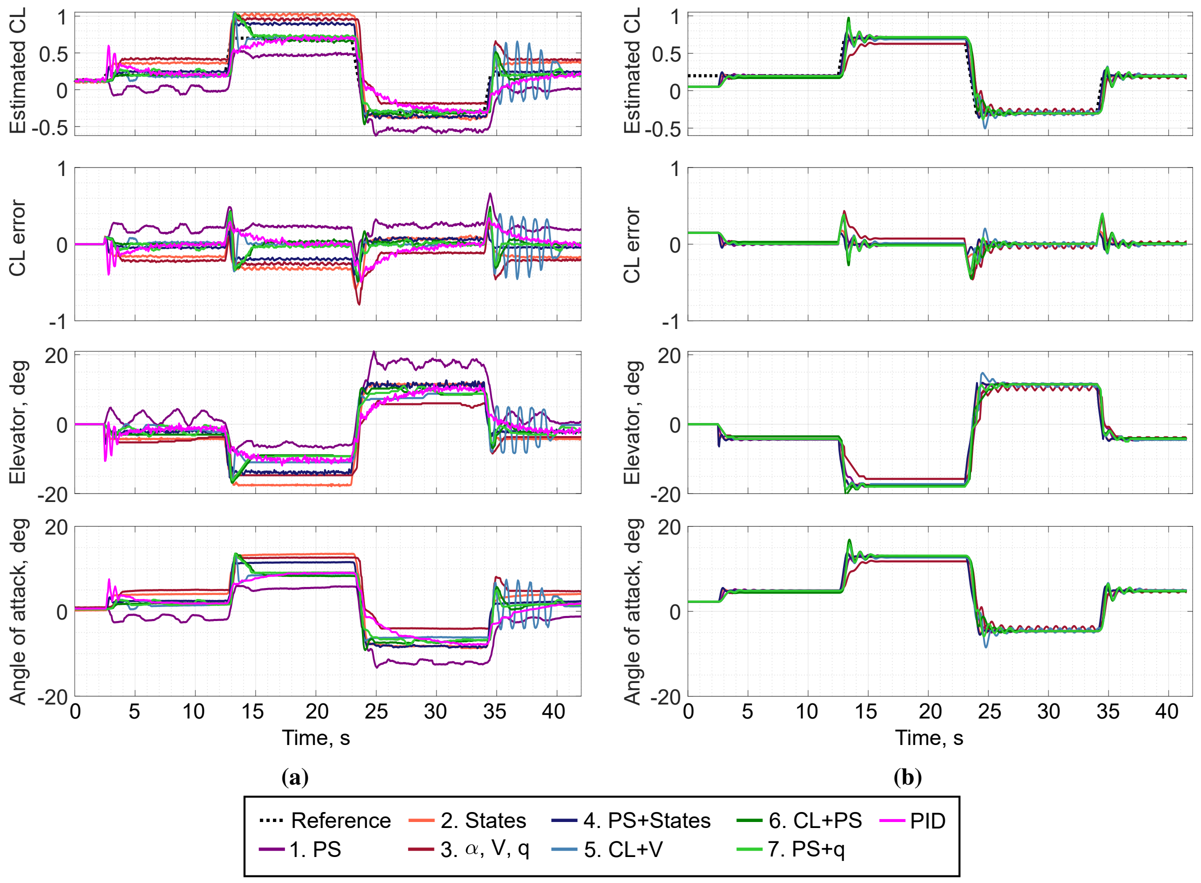


Fig. 7 $\widehat{C}_L$ **tracking time histories at** $V = 15\,\mathrm{m/s}$ **in *(a)* WT and *(b)* simulation tests.**

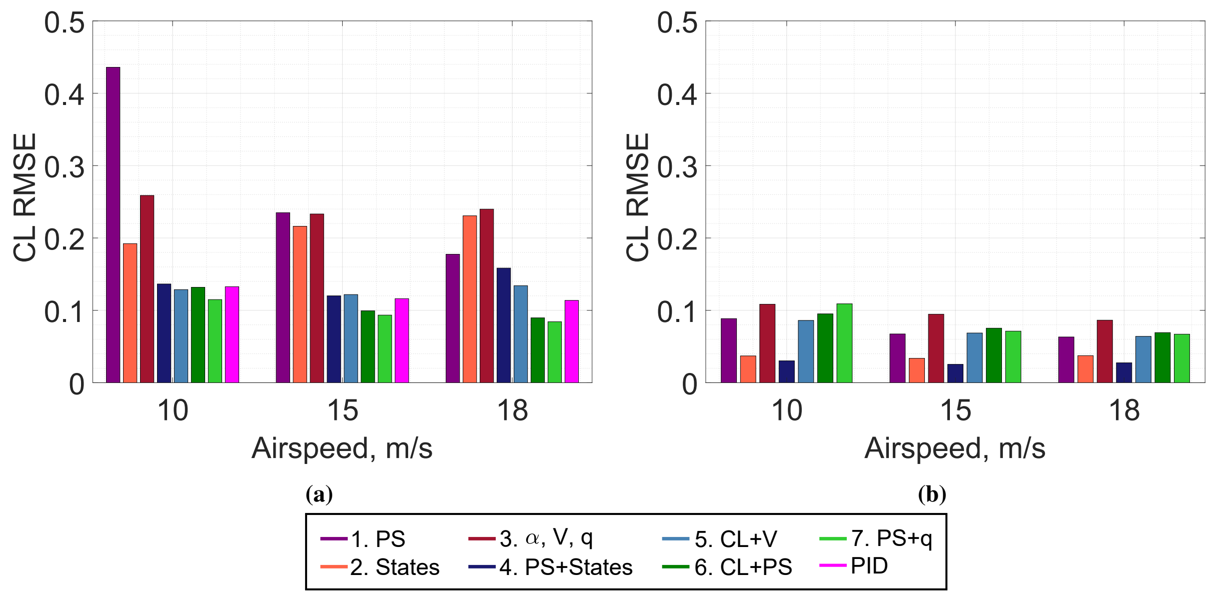


Fig. 8 **Mean** $\widehat{C}_L$ **RMSE values across repeats at different airspeed: *a)* WT, and *b)* simulation tests.**

Overall, the oscillating behaviour was mostly decreased, and the overshoot and overdamping characteristics were increased in the wind tunnel when compared to simulation. This change in behaviour was likely to be a result of minor differences in the dynamics of the simulation model and the physical platform at the time of the validation tests, such as additional friction in the pitch axis caused by setup inconsistencies.

The agents that performed best were the hybrid versions with $PS + q$ information only, given in different forms (6-7). These agents followed the desired signal closely, but presented over and undershoot in the transition phases caused by rapid changes in elevator command as they also did in simulation to a smaller degree. Agent 5 also presented good performance, with low $\widehat{C}_L$ RMSE values overall, but presented oscillating behaviour in some cases.

Agents 1-4 presented the greatest differences between the simulated performance and the wind tunnel tests. The poor performance of Agent 1 in the wind tunnel was attributed to poor convergence during the training. This did not affect its performance when tested in the simulation environment it had learnt from, but given the noise of the sensor readings in WT experiments, Agent 1 resulted in a less robust controller. Agents 2-4 were the only controllers that contained α in their observation space, which was interesting since these agents went to very similar mean α when comparing the wind tunnel results to simulation, and in comparison to the remaining controllers. Conversely, agents 5-7 all presented differences in α achieved when comparing the WT results to those in simulation.

These findings suggested that there was a reality gap between the simulation model and the physical platform which affected the angle of attack. Further study saw an offset between the angle of attack used in the servo-driven dataset described in Section II.A (measured by the servo encoder) and the angle of attack measurement used to build the mathematical pitching moment model and to control the agents in the WT (measured as pitch angle by the IMU). The offset between the measured angles depended on the different test conditions: α, V, q, and was more significant at higher angles of attack. This error in the measurement was attributed to play in the model to rig joint, flexibility of the rig and model, and zeroing of the encoder and IMU angle reference. This meant that the sensor data and the aerodynamic coefficients were misaligned with angle of attack in the simulation model. As a result, the agents which included α in their observation space exhibited different behaviour in the wind tunnel, when the flow conditions at a given α did not match those seen in simulation. The remaining controllers were not directly affected by this reality gap. The relationship between the elevator deflection and the subsequent angle of attack given by the derived pitching moment model was roughly linear, which meant that the effect of the elevator rate on the angle of attack of the platform remained roughly constant throughout the full range of α. In consequence, the simulation model could still see the flow conditions typical of the nonlinear region, but it occurred at a different α than in the wind tunnel. This error in measured α had little impact on the behaviour of those agents that did not use α in their observation space (1, 5-7).

The results presented in Fig. 7a serve as an example of the behaviour of the RL and PID controllers when performing in the linear region. The performance of the RL controllers depended on their observation space, with agents 5-7 performing best overall. The PID controller presented a slow response to changes but was capable of achieving the desired steady state value.

Following on the results of these tests, two of the best performing RL agents (5, 7) and the PID controller were tested at a higher C_L value. The C_{L_d} was defined as the maximum C_L at each tested V, gathered in Table A.4 as described in Section II.E. In this case, the experimental tests were performed with and without perturbation in aircraft configuration (i.e., aileron and flap deflection) to assess the robustness of the controllers.

The time histories of the repeat that produced median $\widehat{C}_L$ RMSE value among the repetitive tests at $V = 15\,\mathrm{m/s}$ are presented in Figs. 9a and 9b for both cases. A summary of the performance of these controllers across the different repeats at this airspeed is presented in Fig. 9c. All controllers reacted to the change in the reference demand, but presented overshoot. The magnitude of the PID overshoot was higher than for agents 5 and 7, and decreased rapidly with a saturated elevator. The PID controller was unable to drive the platform to a desired solution, with $\widehat{C}_L$ RMSE of about 0.2 and a high mean $\widehat{C}_D$, indicative of stall.

Both agents showed good performance, capable of reacting to perturbations, though agent 5 presented the best performance overall with low oscillations in its response to perturbations. This was probably due to the quantity and quality of information in the observation space. This agent had just enough information to solve the problem from a dynamics and aerodynamics point of view: V, $\widehat{C}_L$ and the reference value. Agent 7 had to extract the information corresponding to V and C_L from its inputs, and it presented a less robust performance to perturbations due to the noise in the sensor signals given by the perturbations on the state of the flow around the wing. Nevertheless, the estimated C_L during the tests was mostly under the desired maximum value, and the angles of attack were kept low which meant that the controller presented a conservative solution as opposed to that of the PID controller.

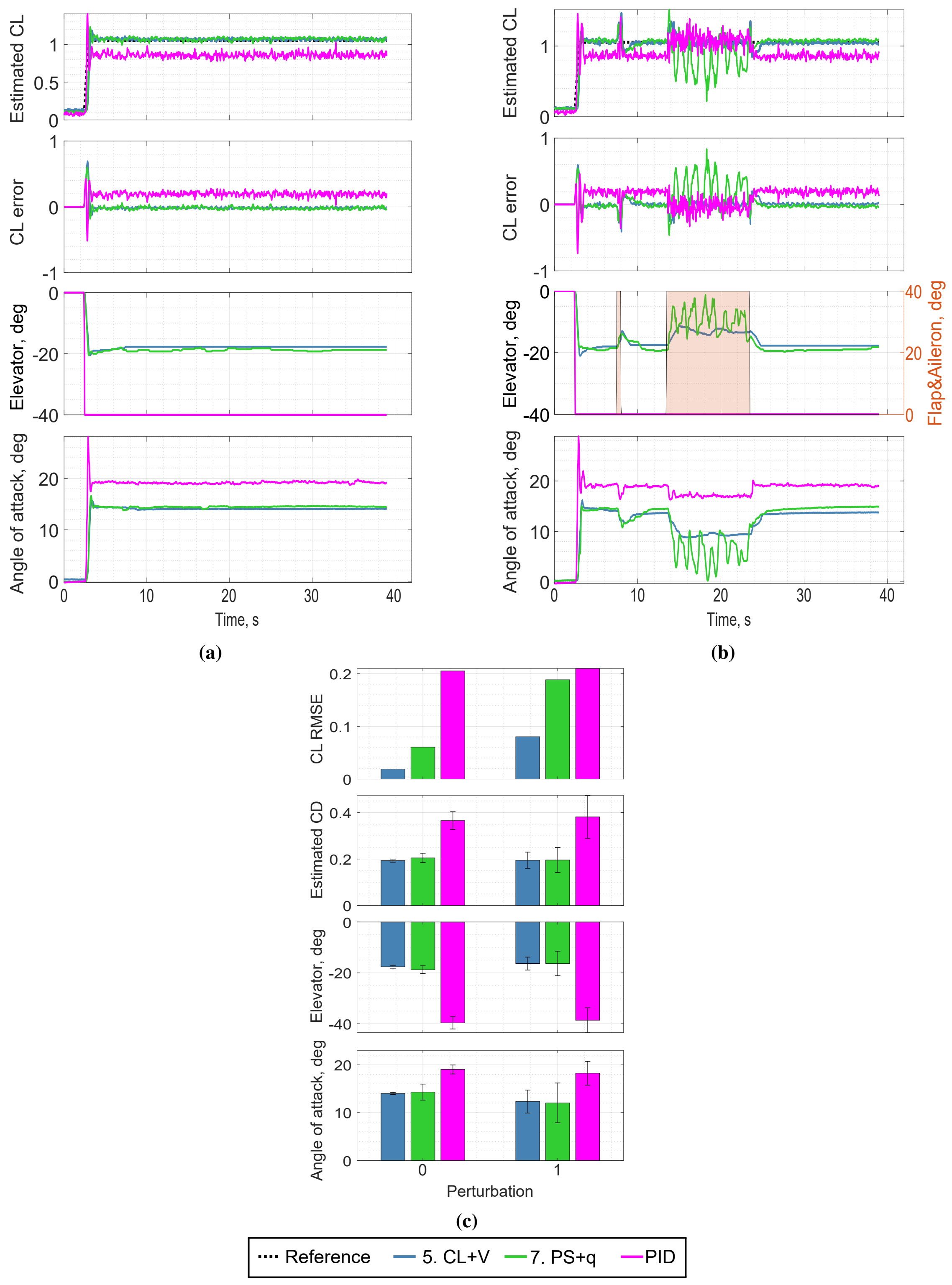


Fig. 9 Agents 5, 7 and PID tracking a desired max C_L at $V = 15\,\mathrm{m/s}$: Time histories with *(a)* no perturbation and *(b)* Perturbation 1; and mean values across repeats for both test cases in *(c)*, where the bars represent one standard deviation. The shaded area in *(b)* corresponds to the period of time in which the aileron and flap were deflected.

V. Discussion

In this work, an empirical 1 DOF flight dynamics model of an aircraft on a pitching rig was built in simulation to replicate the behaviour of the physical platform in the wind tunnel. The simulation model included the information given by an array of distributed pressure and strain sensor readings on the wing. The distributed sensor information was then used to build a C_L and C_D estimator, and to train a series of RL-based controllers to track C_L in simulation. Both the estimator and the agents were tested in WT experiments to validate their performance. All agents together with a manually tuned PID controller tracked a doublet wave demand in C_L, with a maximum $C_{L_d} = 0.7$. These experiments served as a means of testing the performance of the controllers in the linear region. Two of the best performing agents and the PID controller were also tested in the nonlinear region, at higher C_L values as per Table A.4.

A series of discrepancies were visible when comparing the performance of the agents in simulation to their behaviour on the physical platform. In simulation, all agents exhibited similar performance with low $\widehat{C}_L$ RMSE values, which was not surprising given that they were tested in the same simulation environment they were trained on. When tested on the physical platform the simulation model was based on, the performance and behaviour of the agents changed. The differences between the physical and simulated behaviour are likely to be a result of a reality gap between the simulated model and the physical platform, rather than errors in the estimation of the aerodynamic coefficients, as relatively low error values are presented in Section III.

It is important to highlight that the model used for training in this work did not account for noise or uncertainty in the measurements. The ANNs used to estimate both the pressure and strain data, and the aerodynamic coefficients were trained with a noisy dataset, and they found optimal solutions that gave minimal error when tested on a separate testing set. The mathematical pitching moment equation was derived from a noisy dataset, but no further noise was included and the calculated q and α were directly applied in the model. In the same way, V was set as a constant value for each of the tests in the simulated environment whilst the V used in the wind tunnel experiments was measured with a pitot tube instrumented on the wing. The addition of noise and uncertainty on the simulation model could improve the performance of the controllers on the physical platform, helping bridge the reality gap in future work [41, 46, 47].

The readings of pressure and strain sensors alone (agent 1) were enough to control the 1 DOF platform to track $\widehat{C}_L$ in simulation but its performance was deteriorated in the wind tunnel. As was concluded in [22] for V control, the performance of the machine learning controller was enhanced with additional information in the observation space. In this case, this improvement was achieved by combining the distributed sensor readings with V or q. It is interesting to highlight the fact that $PS + q$ was also the information provided to the aerodynamic coefficients estimator in this work. The combination of airflow sensors with IMU and DGPS readings has been suggested in the literature [48, 49] as a means of further exploiting and improving the capabilities of airflow information. The results in this work are in accordance with this.

The combination of PS with q and V as inputs to the observation space led the agents to adapt to the state of the flow instead of relying on specific state values to achieve the demanded C_L. This resulted in some agents (5-7) being particularly robust whilst the agents that contained both α and δ_e in their observation space (2-4), presented rigid behaviours to changes in the model. Their lack of robustness was mainly due to a rigid policy that was highly dependant on the angle of attack or the elevator deflection. This is highlighted in the time histories in Fig. 7. Agents 2-3 strongly relied on both α and δ_e to achieve the optimal solution, even if the state of the flow at the given dynamic state was undesirable. This meant that the "states only" agents did not perform as well as those agents that contained the sensor information in some form.

Agent 4, which contained both the distributed sensor and the states information as inputs made use of both sources of information. At lower angles of attack, the agent was capable of tracking $\widehat{C}_L$ with similar performance to the remaining agents and to simulation, but its performance deteriorated at higher α values. This was in accordance to what was seen in the reality gap study, where the disparity between the two angle of attack measurements was larger at higher α values.

The α reality gap discussed above could have been avoided by either training the sensor estimators with the IMU readings, by using a transformation equation that described the relationship between both α measurements, or by using a multi-hole probe to get α measurements from the wing instead of using the pitch angle reading from the IMU. It is possible that agents 2-4 could see their performance in the wind tunnel improved with these suggested changes. However, their rigid behaviour would still pose a limitation for further application on outdoor flight.

In this work, the PID controller was manually tuned in the wind tunnel for $\widehat{C}_L$ control to serve as a baseline to which to compare RL-based controllers. PID controllers are the most used controllers in commercial autopilots [50], but they are only reliable in a specific range of operation where the relationship between the input and the output is linear. The relationship between the $\widehat{C}_L$ error and the elevator deflection was not linear at high angles of attack, and the PID controller showed poor performance in this region. The comparison presented between the PID controller and the $\widehat{C}_L$ control agents showed the benefits of using RL based controllers for this task, and stressed the limitations of simple linear controllers to control nonlinear systems. It is possible that a different linear approach with a broader range of operating points, such as gain scheduling, would perform better than the PID presented. However, this method would have required an extensive amount of wind tunnel time, and it would still not guarantee good performance around stall. Alternatively, other nonlinear control approaches applied to this problem may also see similar benefits to those presented for RL.

One of the limitations of the control approach presented in this work resides in the fact that the maximum $\widehat{C}_L$ reference value was manually given to the controllers, and that prior knowledge of the actual data was needed for this. It is a question of future research to investigate the use of reinforcement learning as a means to find and control the platform to a maximum C_L unknown a priori. In addition, the overshoot characteristics seen in the control performance of some RL agents suggest that further study into the reward function used in the training should consider a penalisation to account for overshoot. This could be accounted for by limiting the elevator rate of deflection for large changes in C_L error ($C_{L_d} - \widehat{C}_L$).

Overall, the results presented in this work indicated that reinforcement learning was a good approach for lift coefficient control. The PPO algorithm was capable of dealing with the highly complex and large number of inputs, and finding the relationships between the data to optimise the solution to achieve the control task. The hyperparameters used in the algorithm were set to the default values in the Python package Stable Baselines [43]. However, it is possible that different combinations of these may result in better or more robust performance, or in faster learning.

From a practical point of view, the controllers presented in this work could be integrated in a flight controller of an SUAV for longitudinal dynamics control in outdoor flight. One potential implementation of the proposed controller in an SUAV would see the aerodynamic coefficients directly controlled by the elevator, and altitude control achieved with the throttle. This method of SUAV control prioritises the aerodynamic coefficients over attitude. Different missions and applications of distributed sensor systems will see different implementations on a flight controller, but further work on weight optimisation, sensor layout and online learning needs to be carried out in order to ensure a safe and efficient deployment on an SUAV.

The methods presented here could be used in conjunction with manoeuvre trajectory control for agile fixed-wing UAVs [51, 52] to allow for a larger safe flight-envelope to be used, for perched landing approaches to rely on the state of the flow rather than the dynamic state of the aircraft as in [39–41, 53], as a proof of concept to control minimum drag, the onset of stall or other aerodynamic combinations, or for transverse gust alleviation [6–8, 54].

In fact, the use of distributed sensors on the wing for aerodynamic coefficient control gives real-time estimation and knowledge of the state of the flow affecting the vehicle as opposed to model-based methods [6–8]. The work presented in this paper shows that these sensors are capable of providing the necessary information for closed-loop control.

VI. Conclusions

In this work, the experimental readings of an array of pressure and strain sensors distributed across the wing of an SUAV were incorporated into an empirical 1 DOF flight dynamics model. This sensor information was also used to train an ANN to estimate C_L and C_D. The simulation model was used to train a series of RL agents to find policies to control $\widehat{C}_L$ given seven different observation spaces. The RL-based controllers were tested in wind tunnel experiments together with a manually tuned PID $\widehat{C}_L$ controller, which served as a baseline. The performance comparisons gave a better understanding of the effects different inputs and types of controllers have on this task. The results presented serve as a proof of concept for future work. Overall, the findings in this study indicate that:

- The ANN-based aerodynamic coefficients estimators provided accurate estimations (within 1.92%MR for $\widehat{C}_L$ and 2.22%MR for $\widehat{C}_D$) of the coefficients in wind tunnel experiments.
- The distributed pressure and strain sensor readings were enough to track $\widehat{C}_L$ in simulation, but the addition of a pitot tube for V, or an IMU for q were required to achieve the best performance in wind tunnel experiments.
- The agents trained with α data presented rigid policies based on the dynamic states of the platform, which was emphasised by a misalignment in the simulation model that affected their performance in the wind tunnel tests.
- The agents that used PS as an input to their observation space in some form appeared to base their control policy on the state of the flow. This allowed them to better adapt to the discrepancies between the simulation and experimental environments.
- The performance of the agents in the wind tunnel was affected by factors which were not taken in to account in the

dynamics model such as friction in the rig, noise in the load cell or in sensor readings, or rig configuration.

- The reward function could see further improvement by penalising overshoot, which could be done by accounting for large and rapid elevator increases.

Appendix

A 1 DOF pitch dynamics model

$$M = \frac{1}{2}\rho V^2 S c C_m \tag{A.1}$$

$$C_m = C_{m_0} + C_{m_\alpha}\alpha + C_{m_q}\frac{c}{2V}q + C_{m_{\delta_e}}\delta_e \tag{A.2}$$

Table A.1 WOT 4 simulation parameters

Physical constants			
Parameter	Name	Value	Units
g	Gravity	9.81	m/s^2
ρ	Air density	1.225	kg/m^3
Aircraft model parameters			
Parameter	Name	Value	Units
S	Wing area	0.1523	m^2
c	Wing mean aerod. chord	0.254	m

Table A.2 Pitching moment dynamics model for wind tunnel 1 DOF model

V, m/s	C_{m_0}	C_{m_α}	C_{m_q}	$C_{m_{\delta_e}}$
8	4.87×10^{-2}	-1.72	-14.91	-9.19×10^{-1}
10	5.19×10^{-2}	-1.343	-13.38	-7.85×10^{-1}
12	4.41×10^{-2}	-1.11	-12.27	-6.64×10^{-1}
14	3.67×10^{-2}	-9.34×10^{-1}	-11.56	-5.58×10^{-1}
16	3.22×10^{-2}	-8.04×10^{-1}	-11.54	-4.88×10^{-1}
18	2.60×10^{-2}	-6.96×10^{-1}	-10.93	-4.24×10^{-1}
20	2.53×10^{-2}	-5.95×10^{-1}	-10.19	-3.57×10^{-1}

Table A.3 WOT 4 PID gains for $\widehat{C_L}$ control of 1 DOF platform in the wind tunnel

C_L controller			
Parameter	10 m/s	15 m/s	18 m/s
$K_{p_{C_L}}$	−0.5	−0.4	−0.41
$K_{i_{C_L}}$, s	−0.6	−0.3	−0.32
$K_{d_{C_L}}$, 1/s	−0.02	−0.015	−0.001

Table A.4 Maximum C_L values from wind tunnel tests

V, m/s	max C_L value
8	1.10
10	1.08
12	1.07
14	1.06
16	1.05
18	1.04
20	1.03

B Training of reinforcement learning agents

Table B.1 Action space information of selected agents for each task

ID	Observation space	Action space, °/s		
		(1) $\widehat{C}_l$ tracking	(2) $\widehat{C}_l$ max	(3) $\widehat{C}_l/\widehat{C}_d$
X1	PS	[-1, 1] $\times\dot{\delta}_{e_{max}}$	[-1, 1] $\times\dot{\delta}_{e_{max}}$	[-1, 1] $\times\dot{\delta}_{e_{max}}$
X2	States	[-1, 1] $\times\dot{\delta}_{e_{max}}$	[±120, ±50, ±5, 0]	[±120, ±50, ±5, 0]
X3	$[\alpha, V, q]$	[±120, ±50, ±5, 0]	[±120, ±50, ±5, 0]	[±120, ±50, ±5, 0]
X4	PS + States	[-1, 1] $\times\dot{\delta}_{e_{max}}$	[±120, ±50, ±5, 0]	[±120, ±50, ±5, 0]
X5	$\widehat{C}_l + V + [\widehat{C}_d]_3$	[±120, ±50, ±5, 0]	[±120, ±50, ±5, 0]	[±120, ±50, ±5, 0]
X6	$\widehat{C}_l + PS + [\widehat{C}_d]_3$	[±120, ±50, ±5, 0]	[±120, ±50, ±5, 0]	[±120, ±50, ±5, 0]
X7	$PS + q$	[±120, ±50, ±5, 0]	[±120, ±50, ±5, 0]	[-1, 1] $\times\dot{\delta}_{e_{max}}$

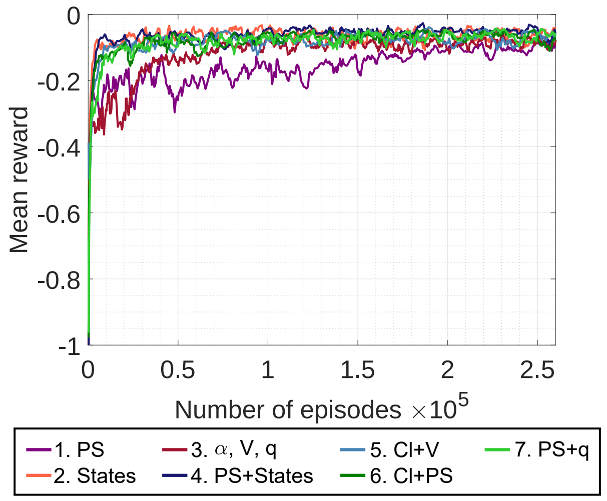


Fig. B.1 Mean reward per episode during training for C_L tracking for the selected agents

Acknowledgements

This work is funded by the EPSRC Centre for Doctoral Training in Future Autonomous and Robotic Systems (FARSCOPE) at the Bristol Robotics Laboratory.

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).

This work was carried out using the computational facilities of the Advanced Computing Research Centre, University of Bristol - http://www.bris.ac.uk/acrc/.

References

[1] Shyy, W., Berg, M., and Ljungqvist, D., "Flapping and flexible wings for biological and micro air vehicles," *Progress in Aerospace Sciences*, Vol. 35, No. 5, 1999, pp. 455–505. doi:10.1016/S0376-0421(98)00016-5.

[2] Watkins, S., Milbank, J., Loxton, B., and Melbourne, W., "Atmospheric Winds and Their Implications for Microair Vehicles," *AIAA Journal*, Vol. 44, No. 11, 2006, pp. 2591–2600. doi:10.2514/1.22670.

[3] Watkins, S., Thompson, M., Loxton, B., and Abdulrahim, M., "On Low Altitude Flight through the Atmospheric Boundary Layer," *International Journal of Micro Air Vehicles*, Vol. 2, No. 2, 2010, pp. 55–68. doi:10.1260/1756-8293.2.2.55.

[4] Watkins, S., Fisher, A., Mohamed, A., Marino, M., Thompson, M., Clothier, R., and Ravi, S., "The turbulent flight environment close to the ground and its effects on fixed and flapping wings at low reynolds number," *5th European Conference for Aeronautics and Space Sciences*, EUCASS, Germany, 2013, p. 10.

[5] Mohamed, A., Poksawat, P., Watkins, S., and Panta, A., "Developing a Stable Small UAS for Operation in Turbulent Urban Environments," *International Micro Air Vehicle Conference and Flight Competition (IMAV)*, 2017, pp. 184–189.

[6] Brunton, S., Dawson, S., and Rowley, C., "State-space model identification and feedback control of unsteady aerodynamic forces," *Journal of Fluids and Structures*, Vol. 50, 2014, pp. 253–270. doi:10.1016/j.jfluidstructs.2014.06.026.

[7] Sedky, G., Lagor, F., and Jones, A., "Unsteady aerodynamics of lift regulation during a transverse gust encounter," *Physical Review Fluids*, Vol. 5, No. 7, 2020, p. 74701. doi:10.1103/PhysRevFluids.5.074701.

[8] Sedky, G., Jones, A., and Lagor, F., "Lift regulation during transverse gust encounters using a modified goman–khrabrov model," *AIAA Journal*, Vol. 58, No. 9, 2020, pp. 3788–3798. doi:10.2514/1.J059127.

[9] Taylor, G. K., and Krapp, H. G., *Sensory Systems and Flight Stability: What do Insects Measure and Why?*, Vol. 34, 2007. doi:10.1016/S0065-2806(07)34005-8.

[10] Brown, R. E., and Fedde, M. R., "Airflow sensors in the avian wing," *Journal of Experimental Biology*, Vol. 179, 1993, pp. 13–30.

[11] Hörster, W., "Histological and electrophysiological investigations on the vibration-sensitive receptors (Herbst corpuscles) in the wing of the pigeon (Columba livia)," *Journal of Comparative Physiology A*, Vol. 166, No. 5, 1990, pp. 663–673. doi:10.1007/BF00240016.

[12] Sterbing-D'Angelo, S., Chadha, M., Chiu, C., Falk, B., Xian, W., Barcelo, J., Zook, J. M., and Moss, C. F., "Bat wing sensors support flight control," *Proceedings of the National Academy of Sciences*, Vol. 108, No. 27, 2011, pp. 11291–11296. doi:10.1073/pnas.1018740108.

[13] Young, J., Walker, S., Bomphrey, R., Taylor, G., and Thomas, A., "Details of insect wing design and deformation enhance aerodynamic function and flight efficiency," *Science*, Vol. 325, No. 5947, 2009, pp. 1549–1552. doi:10.1126/science.1175928.

[14] Shelley, M., and Zhang, J., "Flapping and bending bodies interacting with fluid flows," *Annual Review of Fluid Mechanics*, Vol. 43, No. September, 2011, pp. 449–465. doi:10.1146/annurev-fluid-121108-145456.

[15] Birch, J. M., and Dickinson, M. H., "Spanwise flow and the attachment of the leading-edge vortex on insect wings," *Nature*, Vol. 412, No. 6848, 2001, pp. 729–733. doi:10.1038/35089071.

[16] Sane, S., "The aerodynamics of insect flight," *Journal of Experimental Biology*, Vol. 206, No. 23, 2003, pp. 4191–4208. doi:10.1242/jeb.00663.

[17] Fei, H., Zhu, R., Zhou, Z., and Wang, J., "Aircraft flight parameter detection based on a neural network using multiple hot-film flow speed sensors," *Smart Materials and Structures*, Vol. 16, No. 4, 2007, pp. 1239–1245. doi:10.1088/0964-1726/16/4/035.

[18] Que, R., and Zhu, R., “A Two-Dimensional Flow Sensor with Integrated Micro Thermal Sensing Elements and a Back Propagation Neural Network,” *Sensors*, Vol. 14, No. 1, 2013, pp. 564–574. doi:10.3390/s140100564.

[19] Araujo-Estrada, S., Salama, F., Greatwood, C., Wood, K., Richardson, T., and Windsor, S., “Bio-inspired Distributed Strain and Airflow Sensing for Small Unmanned Air Vehicle Flight Control,” *AIAA Guidance, Navigation, and Control Conference*, American Institute of Aeronautics and Astronautics, Reston, Virginia, 2017. doi:10.2514/6.2017-1487.

[20] Araujo-Estrada, S. A., and Windsor, S. P., “Aerodynamic State and Loads Estimation Using Bioinspired Distributed Sensing,” *Journal of Aircraft*, Vol. 58, No. 4, 2021, pp. 704–716. doi:10.2514/1.C036224.

[21] Araujo-Estrada, S., and Windsor, S., “Artificial Neural Network-Based Flight Control Using Distributed Sensors on Fixed-Wing Unmanned Aerial Vehicles,” *AIAA Guidance, Navigation and Control Conference*, 2020, pp. 1–13. doi:10.2514/6.2020-1485.

[22] Guerra-Langan, A., Araujo-Estrada, S., Richards, A., and Windsor, S., “Simulation of a Machine Learning Based Controller for a Fixed-Wing UAV with Distributed Sensors,” *AIAA UAS Guidance, Navigation, and Control Conference*, American Institute of Aeronautics and Astronautics, Orlando, Florida, 2020, pp. 1–18. doi:10.2514/6.2020-1239.

[23] Que, R., and Zhu, R., “Aircraft Aerodynamic Parameter Detection Using Micro Hot-Film Flow Sensor Array and BP Neural Network Identification,” *Sensors*, Vol. 12, No. 12, 2012, pp. 10920–10929. doi:10.3390/s120810920.

[24] Quindlen, J., and Langelaan, J., “Flush Air Data Sensing for Soaring-Capable UAVs,” *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, American Institute of Aeronautics and Astronautics, Reston, Virigina, 2013, pp. 1–17. doi:10.2514/6.2013-1153.

[25] Wada, D., Araujo-Estrada, S., and Windsor, S., “Unmanned Aerial Vehicle Pitch Control Using Deep Reinforcement Learning with Discrete Actions in Wind Tunnel Test,” *Aerospace*, Vol. 8, No. 1, 2021, p. 18. doi:10.3390/aerospace8010018.

[26] Gu, W., Valavanis, K., Rutherford, M., and Rizzo, A., “A survey of artificial neural networks with model-based control techniques for flight control of unmanned aerial vehicles,” *2019 International Conference on Unmanned Aircraft Systems, ICUAS 2019*, 2019, pp. 362–371. doi:10.1109/ICUAS.2019.8797853.

[27] Ferrari, S., and Stengel, R., “Classical/neural synthesis of nonlinear control systems,” *Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 3, 2002, pp. 442–448. doi:10.2514/2.4929.

[28] Julian, K., Kochenderfer, M., and Owen, M., “Deep neural network compression for aircraft collision avoidance systems,” *Journal of Guidance, Control, and Dynamics*, Vol. 42, No. 3, 2019, pp. 598–608. doi:10.2514/1.G003724.

[29] Chowdhary, G., and Johnson, E., “Adaptive Neural Network Flight Control Using both Current and Recorded Data,” *AIAA Guidance, Navigation and Control Conference and Exhibit*, American Institute of Aeronautics and Astronautics, Reston, Virigina, 2007, pp. 1–21. doi:10.2514/6.2007-6505.

[30] Carcangiu, C., Pujana-Arrese, A., Mendizabal, A., Pineda, I., and Landaluze, J., “Wind gust detection and load mitigation using artificial neural networks assisted control,” *Wind Energy*, Vol. 17, No. 7, 2013, pp. 957–970. doi:10.1002/we.1611.

[31] Collotta, M., Pau, G., and Caponetto, R., “A real-time system based on a neural network model to control hexacopter trajectories,” *2014 International Symposium on Power Electronics, Electrical Drives, Automation and Motion*, IEEE, 2014, pp. 222–227. doi:10.1109/SPEEDAM.2014.6871963.

[32] He, W., Yan, Z., Sun, C., and Chen, Y., “Adaptive Neural Network Control of a Flapping Wing Micro Aerial Vehicle With Disturbance Observer,” *IEEE Transactions on Cybernetics*, Vol. 47, No. 10, 2017, pp. 3452–3465. doi:10.1109/TCYB.2017.2720801.

[33] Kayacan, E., Khanesar, M., Rubio-Hervas, J., and Reyhanoglu, M., “Learning Control of Fixed-Wing Unmanned Aerial Vehicles Using Fuzzy Neural Networks,” *International Journal of Aerospace Engineering*, Vol. 2017, 2017, pp. 1–12. doi:10.1155/2017/5402809.

[34] Gomez, F., and Miikkulainen, R., “Active guidance for a finless rocket using neuroevolution,” *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2003)*, Lecture Notes in Computer Science, Vol. 2724, Springer, Berlin, Heidelberg, 2003, pp. 2084–2095. doi:10.1007/3-540-45110-2.

[35] Sutton, R. S., and Barto, A. G., *Introduction to Reinforcement Learning*, 2$^{\text{nd}}$ ed., MIT Press, Cambridge, MA, USA, 2018.

[36] Ng, A., Kim, H., Jordan, M., and Sastry, S., “Autonomous helicopter flight via reinforcement learning,” *Advances in Neural Information Processing Systems*, 2004. doi:10.1007/978-0-387-30164-8{_}45.

[37] Koch, W., Mancuso, R., West, R., and Bestavros, A., "Reinforcement learning for UAV attitude control," *arXiv*, 2018, pp. 1–13.

[38] Bohn, E., Coates, E., Moe, S., and Johansen, T., "Deep reinforcement learning attitude control of fixed-wing UAVs using proximal policy optimization," *2019 International Conference on Unmanned Aircraft Systems, ICUAS 2019*, 2019, pp. 523–533. doi:10.1109/ICUAS.2019.8798254.

[39] Waldock, A., Greatwood, C., Salama, F., and Richardson, T., "Learning to Perform a Perched Landing on the Ground Using Deep Reinforcement Learning," *Journal of Intelligent and Robotic Systems: Theory and Applications*, Vol. 92, No. 3-4, 2018, pp. 685–704. doi:10.1007/s10846-017-0696-1.

[40] Clarke, R., Fletcher, L., Greatwood, C., Waldock, A., and Richardson, T., "Closed-loop q-learning control of a small unmanned aircraft," *AIAA Scitech 2020 Forum*, Vol. 1 PartF, No. January, 2020, pp. 1–14. doi:10.2514/6.2020-1234.

[41] Fletcher, L., Clarke, R., Richardson, T., and Hansen, M., "Reinforcement Learning for a Perched Landing in the Presence of Wind," *AIAA Scitech 2021 Forum*, , No. January, 2021, pp. 1–14. doi:10.2514/6.2021-1282.

[42] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O., "Proximal policy optimization algorithms," *arXiv*, 2017, pp. 1–12.

[43] Hill, A., Raffin, A., Ernestus, M., Gleave, A., Kanervisto, A., Traore, R., Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., and Wu, Y., "Stable Baselines, GitHub repository," *https://github.com/hill-a/stable-baselines*, 2018.

[44] Jategaonkar, R., *Flight Vehicle System Identification: A Time Domain Methodology*, American Institute of Aeronautics and Astronautics, Reston ,VA, 2006. doi:10.2514/4.866852.

[45] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E., "Scikit-Learn: Machine Learning in Python," *The Journal of Machine Learning Research*, Vol. 12, 2011, pp. 2825–2830.

[46] Jakobi, N., Husbands, P., and Harvey, I., "Noise and the reality gap: The use of simulation in evolutionary robotics," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 929, No. January 1995, 1995, pp. 704–720. doi:10.1007/3-540-59496-5{_}337.

[47] Peng, X., Andrychowicz, M., Zaremba, W., and Abbeel, P., "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization," *Proceedings - IEEE International Conference on Robotics and Automation*, 2018, pp. 3803–3810. doi:10.1109/ICRA.2018.8460528.

[48] Mohamed, A., Watkins, S., Clothier, R., Abdulrahim, M., Massey, K., and Sabatini, R., "Fixed-wing MAV attitude stability in atmospheric turbulence—Part 2: Investigating biologically-inspired sensors," *Progress in Aerospace Sciences*, Vol. 71, 2014, pp. 1–13. doi:10.1016/j.paerosci.2014.06.002.

[49] Gavrilovic, N., Mohamed, A., Marino, M., Watkins, S., Moschetta, J. M., and Benard, E., "Avian-inspired energy-harvesting from atmospheric phenomena for small UAVs," *Bioinspiration and Biomimetics*, Vol. 14, No. 1, 2019. doi:10.1088/1748-3190/aaec61.

[50] Chao, H., Cao, Y., and Chen, Y., "Autopilots for small unmanned aerial vehicles: A survey," *International Journal of Control, Automation and Systems*, Vol. 8, No. 1, 2010, pp. 36–44. doi:10.1007/s12555-010-0105-z.

[51] Levin, J., Nahon, M., and Paranjape, A., "Aggressive Turn-Around Manoeuvres with an Agile Fixed-Wing UAV," *IFAC-PapersOnLine*, Vol. 49, No. 17, 2016, pp. 242–247. doi:10.1016/j.ifacol.2016.09.042.

[52] Levin, J., Paranjape, A., and Nahon, M., "Agile maneuvering with a small fixed-wing unmanned aerial vehicle," *Robotics and Autonomous Systems*, Vol. 116, 2019, pp. 148–161. doi:10.1016/j.robot.2019.03.004.

[53] Maldonado, F., Acosta, J., Tormo-Barbero, J., Grau, P., Guzmán, M., and Ollero, A., "Adaptive Nonlinear Control For Perching of a Bioinspired Ornithopter," *International Conference on Intelligent Robots and Systems*, IEEE, 2020, pp. 1385–1390. doi:10.1109/IROS45743.2020.9341793.

[54] Biler, H., Sedky, G., Jones, A., Saritas, M., and Cetiner, O., "Experimental investigation of transverse and vortex gust encounters at low reynolds numbers," *AIAA Journal*, Vol. 59, No. 3, 2021, pp. 786–799. doi:10.2514/1.J059658.