Article

Modeling and Optimal Control for Rotary Unmanned Aerial Vehicles in Northern Ireland Climate

Jack Gibson and Muhammad Usman Hadi *

School of Engineering, Ulster University, Newtownabbey BT37 0QB, UK; gibson-j25@ulster.ac.uk
* Correspondence: usmanhadi@ieee.org

Abstract: Rotary Unmanned Aerial Vehicles (RUAVs) suffer in average Northern Irish winters due to heavy wind preventing vital tasks from being performed in the economy, such as search and rescue or civil engineering observations. This work provides enhanced stability of RUAVs under wind disturbances by using metaheuristic algorithms to select optimal controller gains. Previous work demonstrated how Particle Swarm Optimization can be used to tune optimal controllers; this work uses a machine learning algorithm (Genetic Algorithm) to tune the controller. Simulations carried out on Full State Feedback, Full State Compensator and Linear Quadratic Gaussian controllers tuned by a variety of techniques revealed that the Genetic Algorithm outperformed conventional manual tuning by 20% and Particle Swarm Optimization by 17% in performance measured in settling time. The proposed method tunes the feedback gains and Kalman filter by Genetic Algorithm, which outperforms the manually tuned conventional schemes and "GA-Hybrid" approach. The conditions required to employ Reinforcement Learning as an alternative method for RUAV stabilization in future scope is also explored.

Keywords: optimal controller; RUAV; particle swarm optimization; genetic algorithm; hybrid control

Citation: Gibson, J.; Hadi, M.U. Modeling and Optimal Control for Rotary Unmanned Aerial Vehicles in Northern Ireland Climate. *Appl. Sci.* **2022**, *12*, 7677. https://doi.org/10.3390/app12157677

Academic Editors: Shanling Dong and Meiqin Liu

Received: 18 June 2022
Accepted: 27 July 2022
Published: 30 July 2022

1. Introduction

The Rotary Unmanned Aerial Vehicle (RUAV) is an increasingly popular vertical take-off and landing (VTOL) aircraft. Low cost, lightweight and agile manoeuvring have contributed to its diverse usage within the Northern Irish economy, from civil engineering observations of silt building on the river Lagan to search and rescue operations on the Mourne mountains and maintenance checks on renewable energy wind farms. Small size is a considerable advantage; however, this has contributed to a major limiting factor, instability caused by wind, which is a significant feature of Northern Irish weather.

1.1. Motivation

Data from Meteorological Aerodrome Reports (METAR reports) from Belfast International Airport indicate that average wind speeds for Belfast during winter months (December–February) for 2021 were 36.5 mph. Named storms such as Storm Eunice gusting up to 60 mph are becoming more frequent [1]. DJI, market leaders in RUAV manufacture, recommend the "MATRICE 300 RTK" for commercial use. This model can safely operate in wind speeds of 33.5 mph [2], it is apparent that the usage of this RUAV would be classified as unsafe by DJI in average Belfast wind conditions during winter months (December–February).

The detrimental impact of this is wide-ranging: in winter months, Belfast City Council are unable to safely operate RUAVs to assess silt levels in the river Lagan, renewable energy companies are unable to perform maintenance checks on windfarms using RUAVS, and search and rescue on the Mourne mountains are unable to use RUAVs to locate missing persons. This is a huge problem which needs to be addressed and is an important area for further research.

1.2. Problem Statement

In order to overcome the issues explained, the current control structure must be reinforced to enable RUAVs to fly in the common high winds experienced in Northern Ireland during winter months. Methods of enhancing current techniques must be explored to create flight controllers that exhibit improved stability than those commercially available. These improvements can be achieved using data-driven methods employing metaheuristic algorithms to select controller gains to stabilize complex systems. Additionally, the nonlinear properties associated with RUAVs are difficult to accurately model and create controllers for, by employing Machine Learning Control to combine with the conventional methods to create a Hybrid approach or to fully replace the conventional methods using Reinforcement Learning.

1.3. Objectives

The main aim of this project is to enhance existing conventional and optimal control structures using data-driven machine learning improving the stability and attitude performance of RUAVs, and shall be explored in the following ways:

1. Conventional PID control with a controller for each of the measured outputs;
2. Optimal Controllers comprising Linear Quadratic Regulator (LQR) used in Full State Feedback configuration, LQR with a Luenberger Observer (LQR + Obs) in a Full State Compensator configuration for state estimation and Linear Quadratic Gaussian (LQG) with Kalman Filter for improved noise attenuation;
3. Enhanced Optimal Controllers using metaheuristic algorithms such as Particle Swarm Optimization (PSO) or Genetic Algorithm (GA) for selecting controller gains;
4. Foundational knowledge required to implement Reinforcement Learning agents.

1.4. Contribution of the Article

The article not only includes and discusses the literature review and current state of the art in depth, but it also encourages the reader to choose a specific approach by outlining the many possibilities that have been used in the past. The article has the following novelties:

1. Firstly, an optimal controller comprising of LQR is used in Full State Feedback configuration, LQR with a Luenberger Observer and LQG with Kalman Filter for noise attenuation enhancing the stability of the RUAVs. Our methodology is novel in a sense that it does not only tune the feedback gain by genetic algorithm, but also the gains of the Kalman filter which outperforms the "GA-Hybrid" approach;
2. Secondly, Metaheuristic Algorithms, namely PSO and GA, are used to optimize the selecting controller gains;
3. Finally, a Reinforcement Learning foundation is outlined, whereby an agent could be used to replace other control structures.

The remainder of this paper is structured as follows: Section 2 covers an extensive review of the state-of-the-art literature. The methodology of each state-of-the-art technique is discussed in Section 3. Section 3.3 is used to explain the current controller structures of Optimal controllers. The fitness function defined in Section 3.4 sets the evaluation criteria for the Section 3.5 (Genetic Algorithm). Tuning both the feedback gains and Kalman filter by a fully data-driven method is novel. Section 4 defines the simulation environment for each controller used. Section 5 reviews the results obtained from each controller tested, with Section 6 covering the future scope of research where foundational knowledge is outlined for using a Reinforcement Learning method using an agent as a potential controller for the RUAV. Finally, Section 7 provides a conclusion to this work. The summary of this article is depicted in the Figure 1, where the overall summary of each respective section is shown.

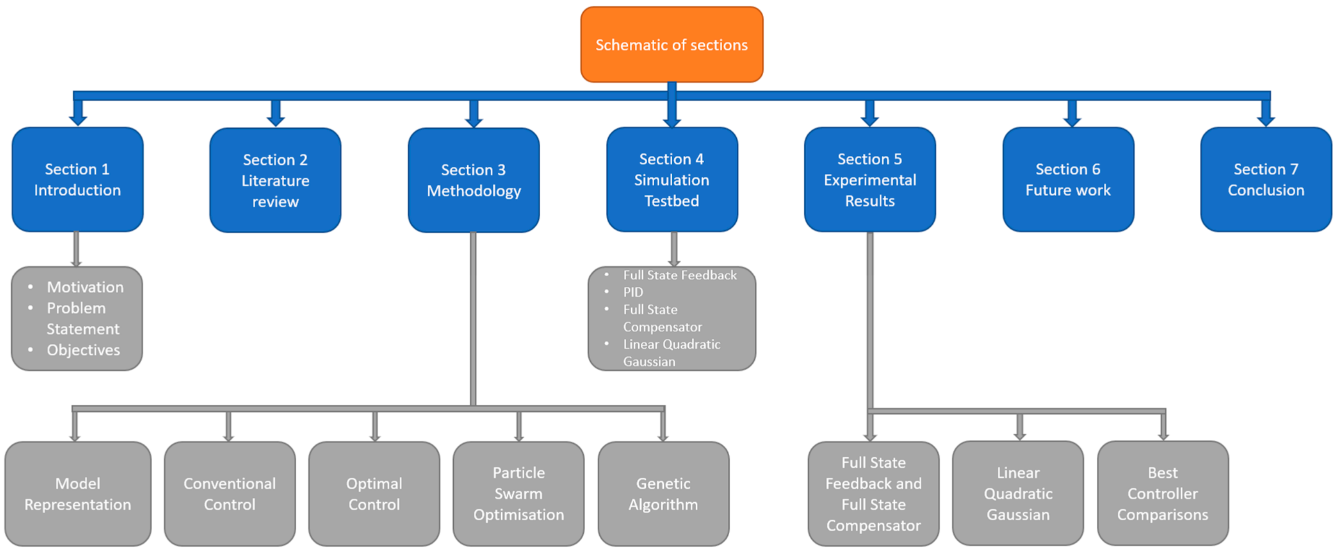


Figure 1. Schematic of sections in this work.

2. Literature Review

In this section, the strategies for controlling RUAVs are discussed, covering a range of Linear, Nonlinear and Artificial Intelligence methodologies. The Linear control methods covers PID control, Linear Quadratic Regulator (LQR), Linear Quadratic Gaussian (LQG) and H infinity Control. The Nonlinear control methods consist of Sliding Mode Control (SMC) and Back Stepping Control. Artificial Intelligence methods include Machine Learning Control, Neural Networks and Fuzzy Logic Control. The diagram summarizing these methods is shown in Figure 2.

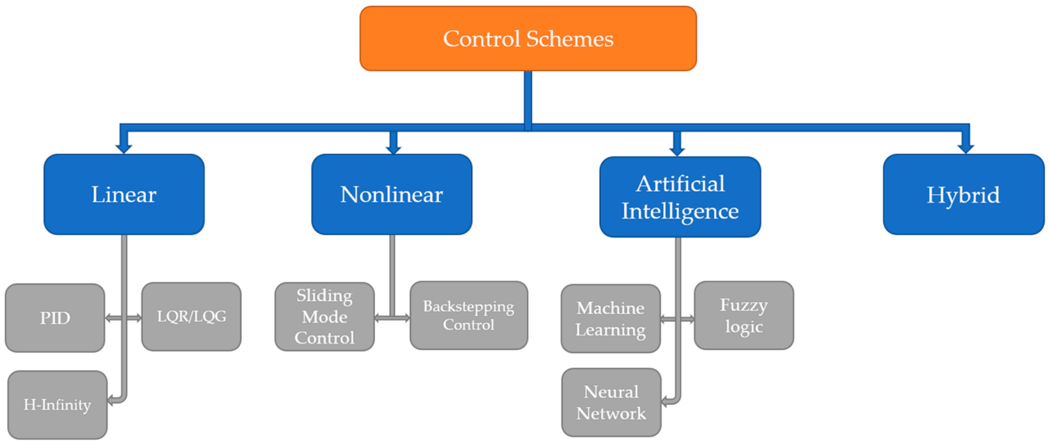


Figure 2. Control Systems Classifications.

The Table 1 abbreviates the control strategies used (PID = Proportional Integral Differential, LQR = Linear Quadratic Regulator, LQG = Linear Quadratic Gaussian, PSO = Particle Swarm Optimization, GA = Genetic Algorithm, AI = Artificial Intelligence).

Table 1. State of the art Control strategies for stability of RUAVs.

No.	Author	Type	Category	Advantages	Disadvantages
1	Zhang et al. [3]	PID Control	Linar	Unit step provides a fast response with small overshoot	Slow and inefficient to retune each time the input changes. No sensor noise attenuation
2	Lahlouh et al. [4]	PID+SOF	Linear	Fast settling time and minimum overshoot when compared to PID	No noise attenuation.
3	Pandy et al. [5]	LQR-TAE	Linear	Better Steady-State Error, transient response and reduction in control effort than PID	No test against external disturbance. Dificult to tune complex systems optimally using the Trial and Error (TAE) method.
4	De-Xin et al. [6]	LQR-TAE	Linear	Controller responds well to disturbance introduced as a sine wave	Sine wave is an unrealistic disturbance. Difficult to optimally tune for a complex system by TAE
5	Chen et al. [7]	Integral LQR-TAE	Linear	Eliminates all Steady-State error when applied to nonlinear system because of the integrator	Disturbances are constant offsets that do not accurately represent RUAV disturbances
6	Baranooni et al. [8]	LQG-TAE	Linear	Gaussian noise in sensor output rejected by Kalman Filter	Complex to tune by TAE for MIMO systems
7	Paponpen et al. [9]	LQG-PSO	Linear	Tuned with a small number of iterations	No improvement to the transient response when tuned with Particle Swarm Optimisation (PSO)
8	Bartys et al. [10]	LQG-PSO	Linear	Lower Control Effort required	No improvement in transient response by tuning with PSO
9	Yu et al. [11]	LQG-PSO	Linear	Better transient response observed for PSO-LQG when compared with PID	Better response is more likely down to comparison of PID with LQG as opposed to LQG and PSO-LQG
10	Walker et al. [12]	H-Infinity	Linear	Improved Anti-Interference when compared to LQR	Slower settling time when compared to LQR
11	Boukhnifer et al. [13]	H-Infinity	Linear	Improved stability in wind disturbance than PID control	With no disturbance H-inf was slower than PID
12	Huang et al. [14]	H-Infinity	Linear	H-Infinity outperforms LQG in high wind disturbances, maintaining low percentage overshoot	Benefits of H-Infinity only become apparent at high winds as LQG was much faster under small disturbances
13	Lec et al. [15]	Sliding Mode Control	Nonlinear	Good rejection of Gaussian noise when used with a sliding mode observer	Control chattering due to high frequency switching
14	Mandeni et al. [16]	Backstepping Control	Nonlinear	If designed in compliance with the Lyapunov stability theorem, stability is guaranteed	Explosion in complexity seen when used on systems with many state. No tests for noise rejection
15	Hwangbo et al. [17]	Reinforcement Learning	AI	Performs well in real applications, highly stable in harsh conditions (RUAV thrownin the air)	High computational cost
16	Waslander et al. [18]	Reinforcement Learning	AI	Model-based method using linear regression instead of Neural Networks, thus, fewer computations	Limited success for a simple step and hover motion
17	Le et al. [19]	Fuzzy Logic control	Hybrid	Impressive Reference tracking as learning rate is updated online	Controller was not tested with any sensor or actuator noise

Table 1. *Cont.*

No.	Author	Type	Category	Advantages	Disadvantages
18	Al-Sharman [20]	Neural Network state estimation	Hybrid	Provides a high level of sensor noise attenuation for use with LQR	Larger level of computational complexity when compared with a Kalman filter
19	Marino et al. [21]	PIDNN	Hybrid	Optimal gains for PID controller improving the conventional Extremum seeking and Zigler Nichols algorithms	No consideration of the practical implementation of the controller, torques possible in simulation may not be feasible in real application
20	Brunton et al. [22]	PID-GA	Hybrid	Genetic Algorithm (GA) used to tune PID gains out-performs Extremum seeking and Zigler Nichols in settling time and overshoot	Implemented on a simple dynamic system so may not transfer accurately to an RUAV.
21	Choubey et al. [23]	LQG-GA	Hybrid	LQG tuned by GA out performs the same controller tuned manually in tracking performance.	Only the Feedback gain element of the LQG was tuned by GA, the Kalman filter was tuned manually.

The main limitation associated with using an optimal controller for real applications is the uncertainty associated with using the trial-and-error method (TAE) for tuning the controller. TAE is a knowledge-based approach used by Pandy et al. [5], De-Xin et al. [6], Chen et al. [7] and Baranooni et al. [8] to select controller parameters, a solid understanding of the system dynamics is required for good performance here. Complex, highly coupled systems become much harder to tune effectively by the TAE method. By using an optimization algorithm to select controller gains, the results obtained by TAE can be improved. The results obtained from Choubey et al. [23] show that the Genetic Algorithm can improve the response of an LQG controller when compared with TAE. However, this method only tuned the feedback gain without considering using optimization for tuning the Kalman filter. Metaheuristic algorithms such as those proposed by Brunton et al. [22], Yu et al. [11] and Bartys et al. [10], such as PSO or GA, can be used to create a controller tuned completely by data driven methods, eliminating the limitations of a knowledge-based approach.

3. Methodology

The proposed controller for providing stability of the RUAV was developed by evaluating different controller structures. The following are the five primary steps taken to realize the best control structure to stabilize the RUAV from harsh wind disturbances.

A. Plant Representation.
B. Conventional Control (PID).
C. Optimal Control.
D. Particle Swarm Optimization application.
E. Genetic Algorithm application.

3.1. Plant Representation

The RUAV model is configured in State Space representation. Due to the complex, highly coupled nature of the dynamic system, the differential equations are arranged in matrix form, making the controller design process simpler. The state space equations are shown in Equation (1) along with the corresponding block diagram for these equations in Figure 3, the matrices for deriving these differential equations can be found in Appendix A.

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{1}$$

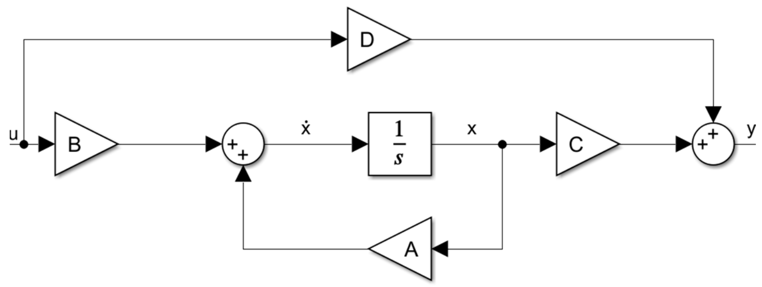


Figure 3. State Space Block Diagram—$\frac{1}{s}$ representing integrator block in MATLAB Simulink.

The A matrix is the system matrix and contains the state information of the system. The RUAV model has eight states—longitudinal velocity (m), normal velocity (w), pitch rate (q), pitch displacement (θ), lateral velocity (v), roll rate (p), roll displacement (φ) and yaw rate (r). The variable x is the state vector containing the current value of the states, in the case of the RUAV model, it is an eight-element vector with each element corresponding to a state in the A matrix.

The B matrix is the Input matrix and describes the behavior of each input to the system. The RUAV has three inputs—pitch displacement (θ), roll displacement (φ) and yaw rate (r). The u variable is the input vector (controller output), u contains the output of the controller, in this case, a three-element vector is required with each element corresponding to the size of the B matrix.

The C matrix is the output matrix and details which of the state are measured as outputs in the system. The RUAV model has five measured outputs—pitch displacement (θ), roll displacement (φ), yaw rate®, pitch rate (q), roll rate (p).

The D matrix is the feedthrough/feedforward matrix, this is used in the system when the system input directly effects the system output. In this case, the RUAV model does not have a D matrix and as such, has a 5×3 matrix of zeros. The y variable details the output of the system.

3.1.1. Output Selection

The RUAV dynamic model can be simplified for some control structures. Three of the input/output combinations are complimentary (pitch displacement, roll displacement and yaw rate), whereas the other two measured outputs (pitch rate and roll rate) are not fed back directly to an input. These two measured outputs can therefore be removed without negatively effecting the system. Removing the extra outputs reduces the hardware required to sense the outputs and effectively stabilize the vehicle where space and weight comes at a premium.

3.2. Conventional Control (PID)

Proportional Integral Differential (PID) control, also known as three-term control, is a simple, yet effective control scheme for systems with no noise. PID controllers are used for controlling Single Input Single Output (SISO) systems, however, the RUAV plant used is a Multiple Input Multiple Output (MIMO) system. To control the system, a multi-loop approach is required where one PID controller is used to control each of the inputs to the system using the three corresponding measured outputs of the same state.

The equation for the controller output (u) is shown in Equation (2) below, the signal error (e) is calculated as the difference in the system output from the reference signal. This error is used in the controller, first by multiplying by the proportional gain, then by taking the differential of the error and multiplying by a differential gain and lastly, by taking the integral of the error and multiplying by an integral gain, these three products are then added together to produce the final output of the controller.

$$u(t) \;=\; K_p e + K_d \frac{de}{dt} + K_i \int_0^t e(t)\,dt \tag{2}$$

The Proportional gain (K_p) is proportional to the current signal error (e), if set too low, the controller will settle slowly, potentially not creating enough controller output to correct the error before the RUAV fails. If set too high, the controller will overcompensate for the error and will oscillate about the setpoint potentially becoming unstable.

The Integral gain (K_i) relates to the speed of response and to totally eradicate steady-state error for accurate asymptotic reference tracking, if set too low, the response will be slow and may not correct the error in time before the system fails, if set too high, the controller will produce a response that overshoots and oscillates about the setpoint.

The differential gain (K_d) works to steady the response when gains for K_p and K_i are set incorrectly causing oscillations. If K_d is set too low, then there may still be oscillations in the system even when at steady state, if K_d is set too high, then the response may be delayed, the desired input may take a long time to reach the output.

3.2.1. PID Control

The block diagram shown in Figure 4 shows the multiloop configuration of the system where a PID controller is used to control the Pitch displacement (θ), Roll displacement (φ) and the Yaw rate (r).

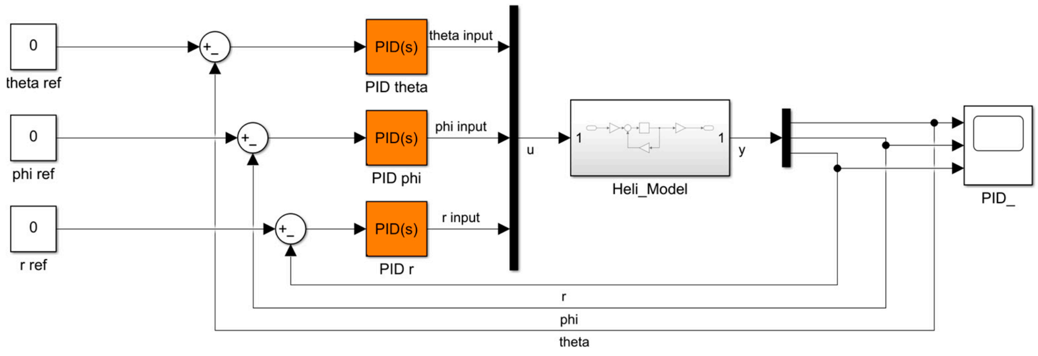


Figure 4. Multiloop PID Control Block Diagram.

The controllers are tuned using MATLAB's Control System tuner, whereby a tracking reference of 20% is defined along with a maximum damping frequency of 10 Hz to prevent excessive oscillation in the time domain in the response and gain and phase margins of 5 dB and 40 deg, respectively. These stability margins were added to ensure robustness to process variations when applied to a real system. The gain margin was set to avoid changes in rotor dynamics/inertia, while the phase margin was introduced to avoid problems introduced with time sensitive slow computational hardware.

3.2.2. PID Control with Static Output Feedback

PID control alone can struggle to achieve constant zero steady-state error quickly, in multivariable systems; to aid this, a Static Output Feedback (SOF) gain was added [4]. To implement this, the removed measured outputs discussed in Section 3.1.1 were added back to the system to offer the SOF gather more information on the state of the dynamic model to improve the stability of the RUAV. This method eliminates all steady-state error and oscillations from the system. The SOF gain is tuned using the same tuning rules defined for the PID controllers in Section 3.2.1; the block diagram for this control structure is seen in Figure 5.

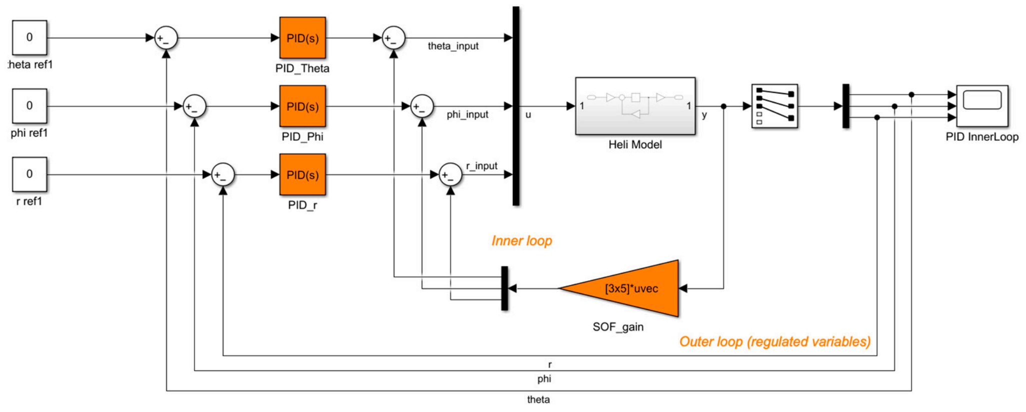


Figure 5. Multiloop PID with Static Output Feedback Control Block Diagram [24].

3.3. Optimal Control

Optimal control strategies aim to extract the best performance from a dynamic system for the minimum cost. Several methods have been used to create the controller have been explored. Section 3.3 is used to explain the current controller structures of optimal controllers, the authors do not claim this section to be novel. This section was added to provide a basic underlying knowledge required for better readability for the wider audience.

3.3.1. Full State Feedback (Linear Quadratic Regulator)

The Full State Feedback controller assumes that every state in the system is available for use in the controller. The input defined as u in Equation (1) is calculated for Full State Feedback with:

$$u = -Kx \quad (3)$$

The K variable is the gain of the Full State Feedback controller. Input to the system (u) is calculated as the product of the negative feedback gain and the state vector (x). The controller architecture of this controller is defined in Figure 6 [25].

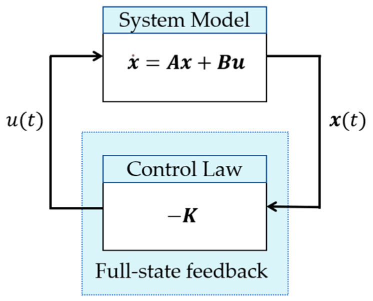


Figure 6. Full State Feedback Controller.

To equate the controller gain K in Equation (3) for LQR a series of procedures must be followed:

1. Define the Q and R matrices;
2. Solve for S in the Algebraic Riccati Equation;
3. Compute the controller gains K from generated S;
4. Evaluate produced controller gains to select the optimal controller.

The Q and R matrices are square matrices the size of the state and input vectors used for weighting the cost function. Each diagonal element of the Q and R matrices corresponds to the element of the State or Input vector that they penalize. The cost function J shown in Equation (4) is used for measuring the cost of the controller. Although the variable J is not explicitly used in Equations (5)–(8), it shows the relationship between speed of response relating to the Q matrix and the actuator effort in the R matrix used to achieve the desired response from the controller.

By setting elements in the Q matrix high, the non-zero states in the $\overline{x}^T$ vector brought about by wind disturbance will introduce a high cost in Equation (4), prioritizing the offset state for faster return to equilibrium. By setting elements in the R matrix high, the cost of using an actuator is increased, meaning that the controller will prioritize return to equilibrium using small actuator effort. For fast return to equilibrium where actuator effort is not a concern, elements in the Q matrix are set high, while elements in the R matrix are set low; this produces a greater cost for non-zero state values. Contrastingly, if the speed of correction is not a priority but rather, conservation of actuator effort is mandatory, then values in the Q matrix are set low with values in the R matrix set high; this produces a greater cost for using actuators to respond to the state offset.

$$J = \int_0^\infty \overline{x}^T.Q\overline{x}(t) + \overline{u}(t)^T R\overline{u}(t)\, dt \tag{4}$$

Once the Q and R matrices are defined, the second step is to solve for S in the Algebraic Riccati Equation defined in Equation (5). This produces more than one solution, however, only one will stabilize the system optimally. The Q and R matrices used in Equation (4) defining the cost are used in Equation (5) as the next step towards solving controller gain K.

$$A^T S + SA - SBR^{-1}B^T + Q = 0 \tag{5}$$

Once S is found, the feedback gain K can be computed using Equation (6) below.

$$K = R^{-1}B^T S \tag{6}$$

As there is more than one solution for Equation (5), there are many controllers created from Equation (6), each should be evaluated to identify which is optimal. To evaluate the produced controllers, the closed loop eigenvalues are generated using Equations (7) and (8). The controller that pushes all eigenvalues of the closed loop system into the negative domain is taken as optimal.

$$CL = A - B.K \tag{7}$$

$$Eigenvalues = \det(CL - \lambda\, I) \tag{8}$$

The optimal controller gain K generated and evaluated by Equations (5)–(8) can then be used as part of Equation (3) as the feedback term generating the input u to bring about stability in the system.

3.3.2. Full State Compensator

Full State Feedback measures each state using sensors. These sensors often contribute to Full State Feedback's main shortcoming; its large hardware requirement to measure each of the internal states of the RUAV. This is not only expensive in terms of hardware, but is also computationally expensive for the controller to have constant feedback from all states of the system (eight sensors required in the case of the RUAV model).

Instead, the values of the states of the system are estimated so less emphasis is placed on highly accurate, constant sensor data. The Full State Feedback gain is used with a Luenberger Observer predicting the internal states of the system to create a Full State Compensator. The controller architecture of the Full State Compensator is shown in Figure 7 [25]. The main change from Full State Feedback is that the feedback gain K is multiplied by the predicted state $\hat{x}$ from the Observer instead of the actual internal states of the system. By using the Luenberger Observer, a smooth transient response can be achieved even when the real internal states may be compromised. This is particularly applicable to Aerial Vehicles where weight comes at a premium, by using an observer component, the internal states of the vehicle do not need to be measured by bulky sensors.

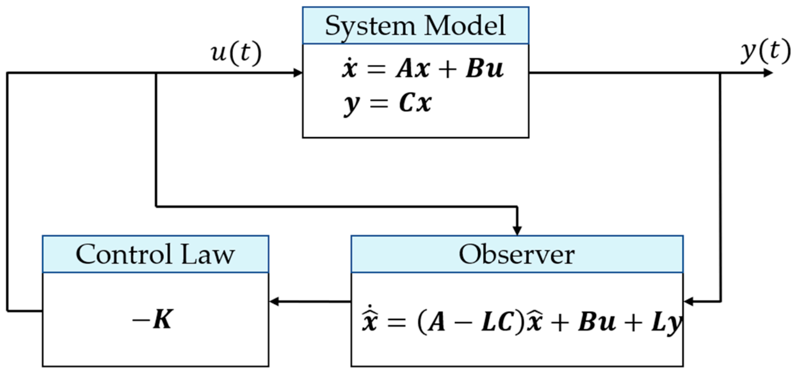


Figure 7. Full State Compensator.

The State Feedback gain is calculated using the same process as the Full State Feedback discussed in Section 3.3.1. The Observer gain is calculated in the same four steps as Full State Feedback with slight modifications. The A and B matrices are replaced with transposed A and transposed C matrices while the Q matrix comprises the product of B and B transposed matrices. The R matrix remains the same as the Full State Feedback.

3.3.3. Linear Quadratic Gaussian (LQG)

While the Luenberger Observer works well in instances where all states cannot be measured, those states that are measured must be noiseless if an acceptable response is to be achieved. However, this is not usually the case, as even expensive sensors can inherit some measurement noise, many dynamic systems also suffer from process noise from high frequency actuators (such as motors used to spin RUAV rotors). The Linear Quadratic Gaussian controller provides the same benefits of a Luenberger Observer, but with the added advantage of improved noise attenuation. The controller architecture is shown in Figure 8 below [21].

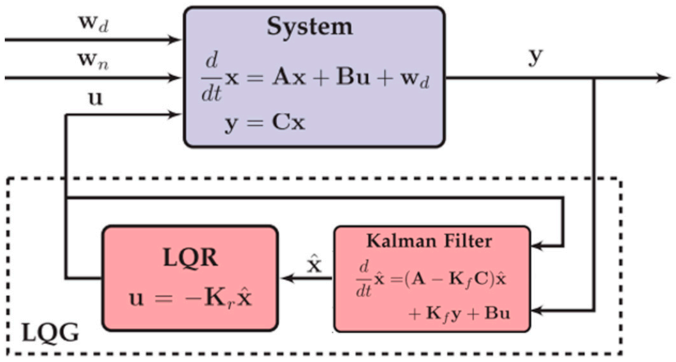

System
$\frac{d}{dt}x = Ax + Bu + w_d$
$y = Cx$
w_d
w_n
u
y
LQR
$u = -K_r\hat{x}$
$\hat{x}$
Kalman Filter
$\frac{d}{dt}\hat{x} = (A - K_f C)\hat{x} + K_f y + Bu$
LQG

Figure 8. Linear Quadratic Gaussian.

The input u to the system is the same as for Full State Feedback discussed in Section 3.3.1, but the Kalman-estimated $\hat{x}$ is used instead of all measured states. The input u and output y are then fed back into the Kalman filter to reject Gaussian noise.

The LQG controller is tuned similarly to the LQR with Luenberger Observer, the Feedback gain K is tuned in the same way. The Kalman filter is tuned slightly differently, instead of a Q matrix, there is a Vd (process covariance matrix) matrix, the same dimensions of as the Q matrix. Tuning Vd requires a knowledge of where the noise is in the system, where each diagonal element in the matrix corresponds to the location of the noise in the system (e.g., Vd (1, 1) corresponds to noise in State (1). The Vn (sensor covariance matrix) matrix, similar to the R matrix in LQR controller, sets the control effort. If the value is set low, the transient response settles very quickly, but with noise in at steady state. The noise disturbance to the system is assumed to be zero-mean Gaussian. Figure 9 shows how isolating and increasing the values in the Vd matrix the shape of the transient response changes. At low values, the response is very slow but rejects all noise, whereas Vd at higher values produces a much faster response but is noisy. Comparatively, if Vn is set high, the response is slower, but more noise is rejected from the system. Tuning the parameters of Vn and Vd is a trade-off between aggressiveness of signal estimation and noise attenuation, this tuning process can be difficult for multivariable systems with many states.

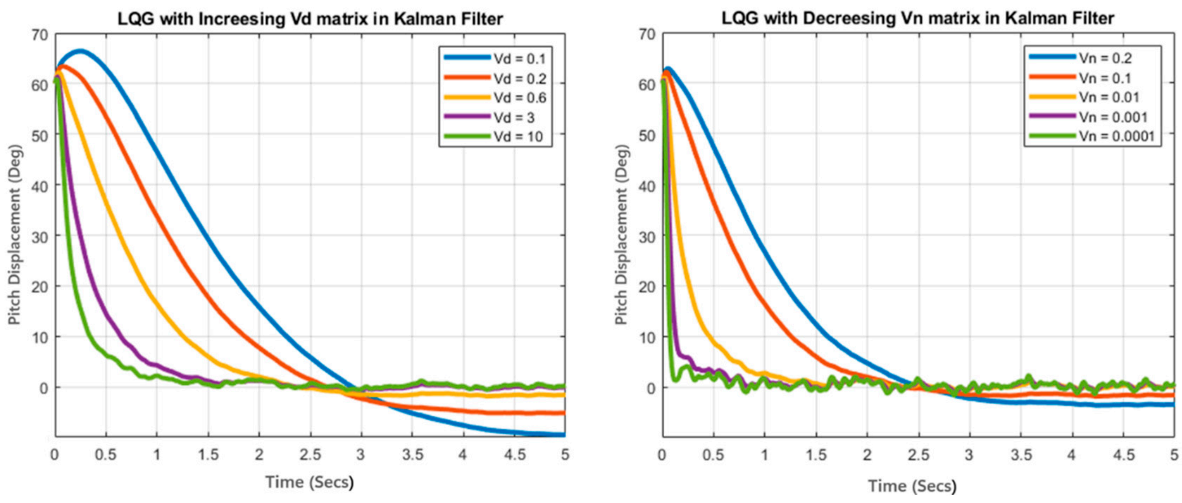


Figure 9. Comparison of varying Vd and Vn matrix values.

The Kalman gain K_f, as seen in Figure 8, can be obtained using the same process as used in Sections 3.3.1 and 3.3.2. The Q and R matrices are swapped for the Vd and Vn matrices while the A and B matrices are swapped for A transposed and C transposed.

3.4. Particle Swarm Optimization Algorithm

The Particle Swarm Optimization, also known as PSO, is a heuristic algorithm that can be used to tune the matrices of a controller based on maximum and minimum user-defined values. This method has been detailed by many authors as shown in Table 1. The inspiration for this algorithm comes from the behavior of flocks of birds or schools of fish. The number of particles (individuals) is first defined, the larger the number of particles, the more likely the swarm is to find the global optimum in the set number of iterations. The maximums and minimums for the values to be tuned are held within the W max and W min variables. The W variable represents the inertia weight and is the distance the particle can search within an iteration.

The C_1 and C_2 values relate to the particles individual and social behavior respectively. If C_2 is set high, there will be a high level of "communication" between the particles in the swarm. Based on Equation (9), a greater emphasis will be placed on the personal best over the group best. The swarm will converge quickly, but possibly on a local minimum instead of the global minimum. If C_1 is set high, then the emphasis is placed on the individual in the swarm without communication between particles, the exploration space is large, but the particles do not work as a "team" to find the global minimum. Balancing the C_1 and C_2 values is key to the success of the algorithm. The elements for r_1 and r_2 introduce randomness into the calculation to change the result each time. P_{best} and G_{best} correspond to the particles optimum and group optimum, respectively. The calculation for v_i below uses these values to derive the velocity of the $i'th$ particle in the swarm, then update its position x using its previous position and the newly calculated velocity.

$$v_i = Wv_i + C_1 r_1 (P_{best,i} - x_i) + C_2 r_2 (G_{best} - x_i) \tag{9}$$

$$x_i = x_i + v_i \tag{10}$$

A Fitness Function is required for testing the gains chosen by the algorithm, this can be configured as a cost function, better gains will result in a drop in the overall cost function of the controller. The Fitness Function used is Eigenstructure Assignment, where the eigenvalues of the devised controller in the closed loop system are taken, as shown in Equation (11), the observer component (Lo.C) is removed for Full State Feedback controller. This fitness function was selected as it incorporates both eigenvectors and eigenvalues in the equation relating to the speed and shape of response respectively directly correlating to the settling time and % overshoot of the closed loop system. With each iteration of the algorithm, the poles of the closed loop system are moved further into the negative domain, stabilizing the system.

$$[V, D, W] = \det((A - B.K + Lo.C) - \lambda\ I) \tag{11}$$

The left and right eigenvalues (V, W) are taken to evaluate S in Equation (12). In Equation (13), the absolute sum of S is taken as the fitness value, large values show a poor performance of the algorithm while small values show good performance.

$$S = \frac{W^T V}{||W||_2 ||V||_2} \tag{12}$$

$$J = \left|\sum S\right| \tag{13}$$

The code for the calculation of optimal values using PSO algorithm is defined in Pseudocode 1.

Pseudocode 1: Particle Swarm Optimisation

```
Initialize the control parameters (N, c1, c2, Wmin,     Wmax  and MaxIter)
Initialize pop of N particles
for 1: MaxIter
do
    for each particle
        Calculate the objective of the particle based on Fitness Function
        Update P_best if better than previous
        Update G_best if better than previous
    end for

    Update the inertia weight (W)
    for each particle
        Update the velocity (v)
        Update the position (x)
    end for


    Return G_best as the best estimation of the global optimum

end for
```

3.5. Genetic Algorithm

The Genetic Algorithm (GA) is another metaheuristic algorithm used for locating optimal gains for controllers. The inspiration for this algorithm comes from the theory of evolution and natural selection where the best individuals in an environment will pass genes onto the next generation, compounding into a highly evolved individual well suited to a given task. The process used in GA can be used to tune controller parameters completely from scratch or adapt and improve those selected manually.

Firstly, the number of individuals is defined making up the total population along with the maximum number of generations. Usually, the larger the population and the number of generations the more likely the algorithm will find the global minimum. However, creating a bigger population and maximum generation value requires more computational effort making the algorithm slower to run on suboptimal hardware.

The population works following a set of predefined 'genetic rules' to minimize a cost function (Fitness Function) producing optimal controller gains for K_r (feedback gains) and K_f (Kalman gains). Each individual in the population is made up of a defined number of chromosomes. The number of chromosomes relates to the values of the matrices to be tuned. For example, a Full State Feedback controller used to tune a system with 3 inputs and 8 states will require a Q matrix of size 8 and an R matrix of size 3. The resulting individual will require 11 (8 + 3) chromosomes to tune the controller.

Each chromosome is initially set to a random value for each individual if the algorithm is tuning from scratch. Alternatively, if the algorithm is improving a controller from previous approaches such as manual tuning the values of the control matrices are to be used as the starting point for 60% of the population with the other 40% being set to random to encourage exploration of new values. Each individual is then evaluated using the Fitness Function and ranked on its cost.

There are 5 genetic rules to be followed:

1. Elitism—The top 5% of the population advances to the next generation totally unchanged.
2. Replication—The top 5% of the population advances to the next generation and is replicated 19 times to make up 95% of the new population, this combined with
3. The Elite population makes up the total population. These individuals are modified using genetic rules 3 and 4.
4. Crossover—Two individuals are selected and are snipped at a defined section of their chromosomes; these snipped sections are swapped between the individuals. This process is used to exploit and enhance the existing successful strategies found by the algorithm.
5. Mutation—Chromosomes within the individual are changed to a random value. This promotes diversity in the population and increases the scope for exploration of new values.

The new population is then evaluated on the same Fitness Function, as described in Section 3.4, Equations (11)–(13). The resulting costs are used to rank each individual, the new best 5% advances to the next generation. Steps 1–4 are repeated, and the evolution process continues until the maximum generation is reached where the algorithm will print the global optimum. The basic code used for the Genetic Algorithm along with the block diagram can be found in Pseudocode 2.

Pseudocode 2: Genetic Algorithm

```
Initialize the control parameters (PopNum, ChromNum, CrossVal,
                                   MutVal, MaxGen)
Gen = 1
for 1: MaxGen
do
    for   each individual
        Calculate the cost of the individual based on Fitness Function
    end for

    Rank individuals based on cost into ranked_pop
    Advance top 5% of ranked_pop and copy to match PopNum

    for 1: 0.95*new_pop
        Crossover snip and cross at CrossVal
        Mutate the chromosome MutVal
    end for

    Return Lowest Cost
    Return Gen

    Gen = Gen+1
end for
```

4. Simulation Environment

To validate the proposed algorithms as an improvement to traditional methods for tuning controllers, a simulation environment was developed. This environment contains each of the optimal control structures outlined in Section 3.3, along with the benchmark PID controller. Each of the optimal controllers was tuned using three methods, Manual Tuning, Particle Swarm Optimization and Genetic Algorithm. For those with an observer component (Full State Compensator and LQG) Genetic Algorithm—Hybrid was added as a fourth tuning option, whereby the observer was tuned manually with the feedback gain being tuned using GA. A graphical representation of the simulation environment is shown in Figure 10.

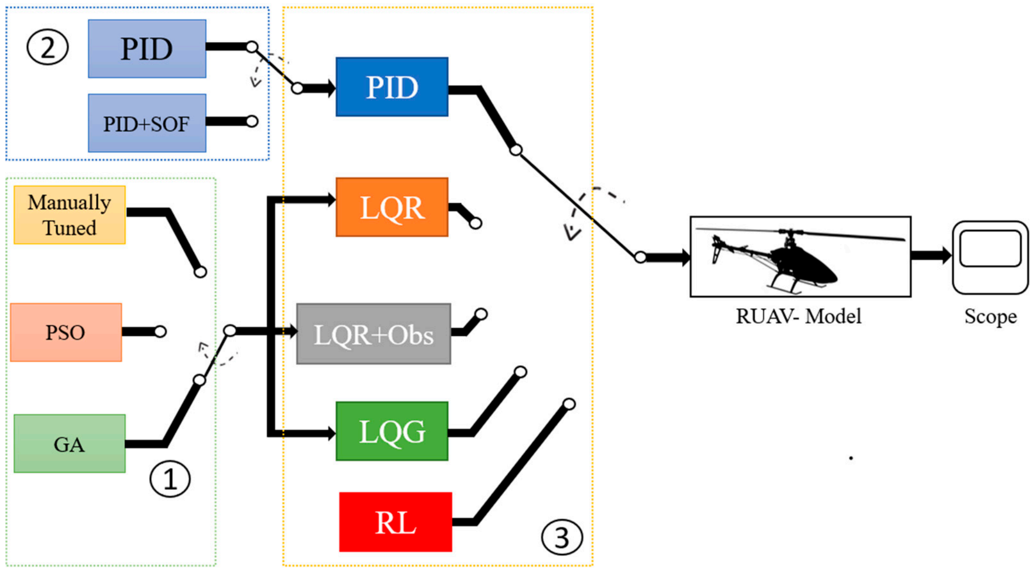


Figure 10. Simulation Environment for RUAV control schematic. The functions are as follows: 1: Allows for switching between different tuning methods for the selected controller. 2: Allows for switching between classic PID and PID controller with Static Output Feedback to enhance response. 3: Allows the vehicle model to be configured to each of the controllers where their responses can be analyzed in the scope.

Tuning Parameters for Each Controller Type

The table in Appendix B shows the parameters used in simulation of the Full State Feedback controller along with the tuning gains and static output feedback gains for the PID benchmark. Initial Conditions are applied to the State matrix to generate an offset in the model, initializing simulation outside the equilibrium condition. Process and measurement noise coefficients are also defined to test noise attenuation for each controller architecture. Appendix B also details the Q and R matrices used in manual tuning, along with the hyperparameters used in both the Particle Swarm Optimization and the Genetic Algorithm. Finally, the resulting feedback gains (K) used in the Full State Feedback controller are defined for each tuning method. The initial conditions and noise parameters remain constant throughout simulation and as such are identical to those defined in Appendices C and D.

Appendix C shows the parameters for tuning the Full State Compensator. The Q and R matrices are detailed for both the feedback gain and the Luenberger observer gain. The feedback gain for manually tuned remains the same as manually tuned Full State Feedback gain which are detailed in Appendix B.

The table in Appendix D contains the parameters for tuning the Linear Quadratic Gaussian. The Q and R matrices are detailed for tuning the feedback gain while the Vd and Vn matrices are defined for tuning the Kalman gains for the manually tuned method. The hyperparameters for generating controller gains using the Genetic Algorithm and Particle Swarm Optimization are also defined.

Each of the controller architectures used in the simulation environment are defined in block diagram form in Appendix E. The feedback gains and Luenberger/Kalman gains are altered depending on the simulation configurations defined by Figure 10. All other gain blocks are used to represent the configuration of the Observer or the RUAVs dynamic model these are to remain the same throughout the simulation process.

5. Simulation Results

The findings from the simulation testbed described in the previous section are explained here. The two methods of evaluating the performance of the devised controllers are by using percentage overshoot (%OS) and settling time (Ts). These two criteria are used as the %OS measures how much the vehicle overshoots its hover state, ensuring %OS is as low as possible is key to the success of the selected controller. Additionally, the Ts is key as it ensures that the vehicle is within the equilibrium boundary (hover state) in the quickest time possible. Noise attenuation will be evaluated using the generated plots in Figures 11–14. Each contains the response of controllers' three measured outputs, Pitch displacement (θ), Roll displacement (φ) and Yaw rate (r) with an added integral of Yaw rate giving Yaw displacement.

5.1. Full State Feedback vs. Full State Compensator

The responses for Full State Feedback and Full State Compensator are shown in Figures 11 and 12. The response produced by the Full State Feedback controller shows that all three tuning methods are heavily compromised by the noise in the system, this is due to the architecture of the controller as opposed to a specific tuning method.

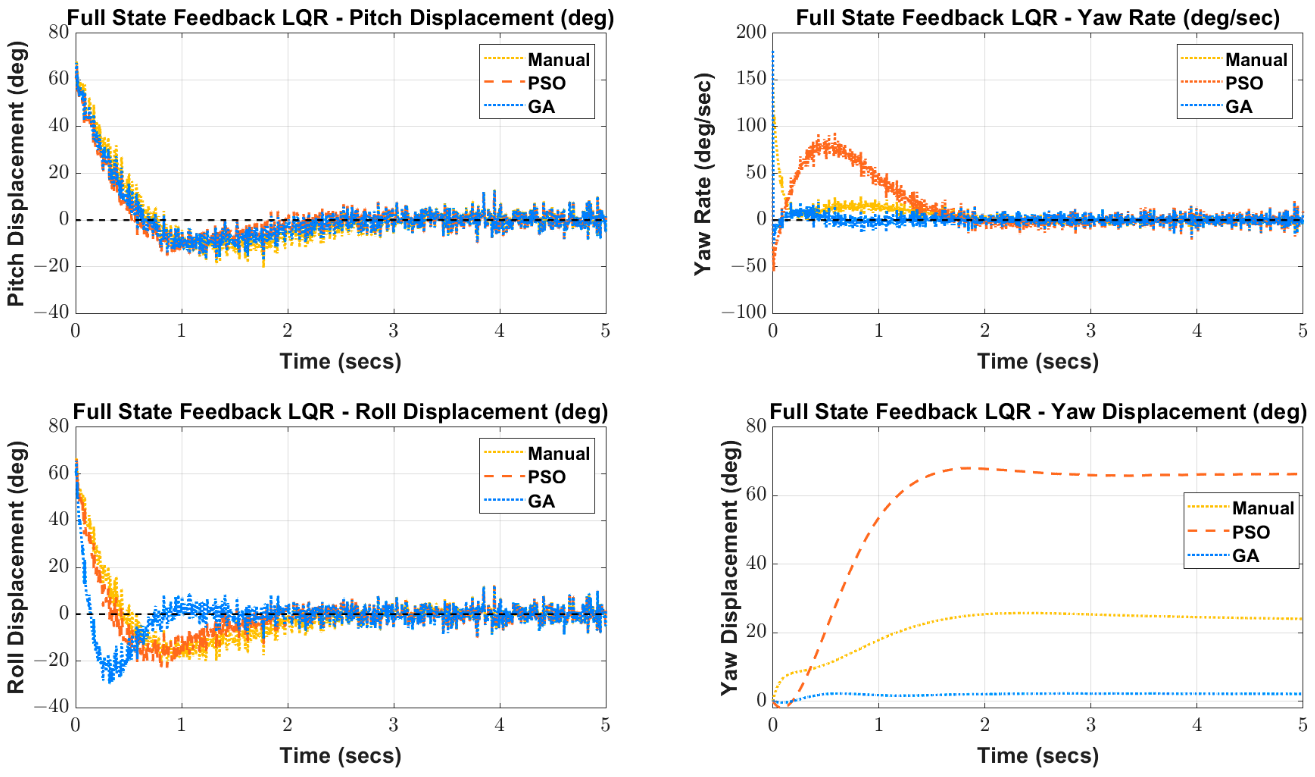


Figure 11. Full State Feedback response for Manually tuned, PSO and GA.

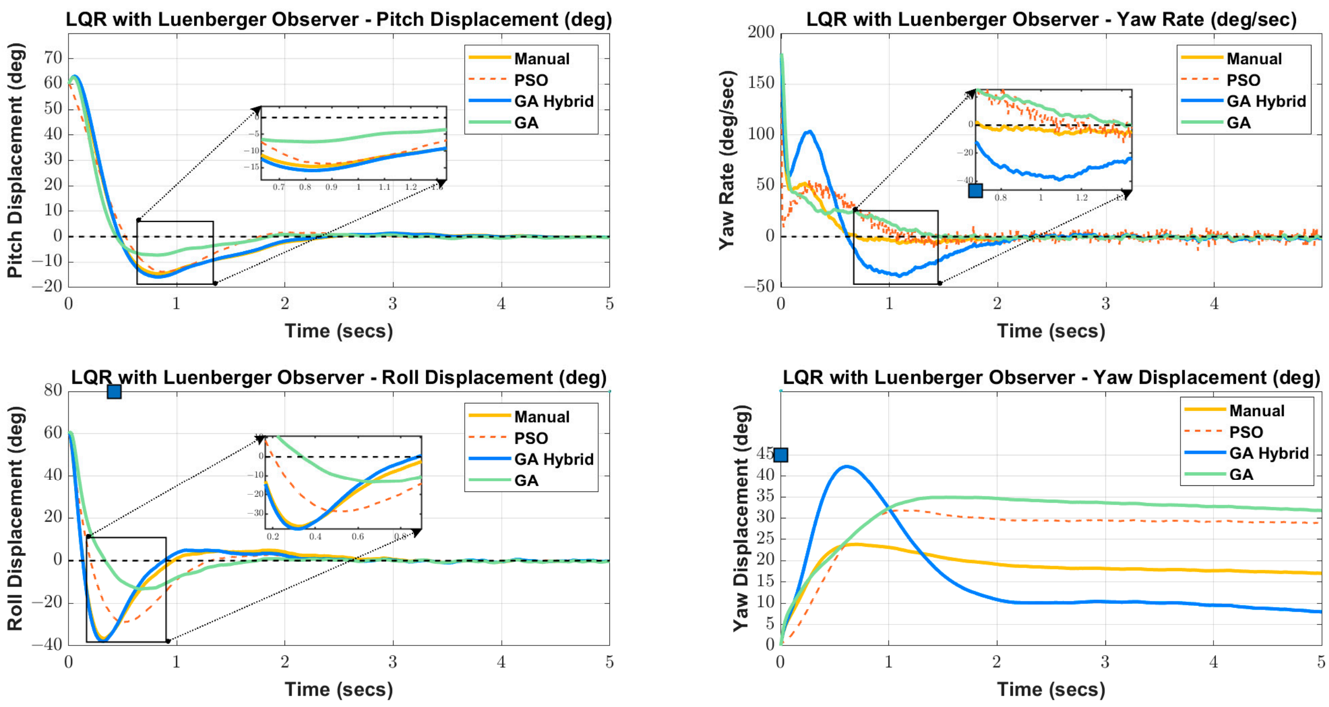


Figure 12. Full State Compensator response for Manually tuned, PSO and GA.

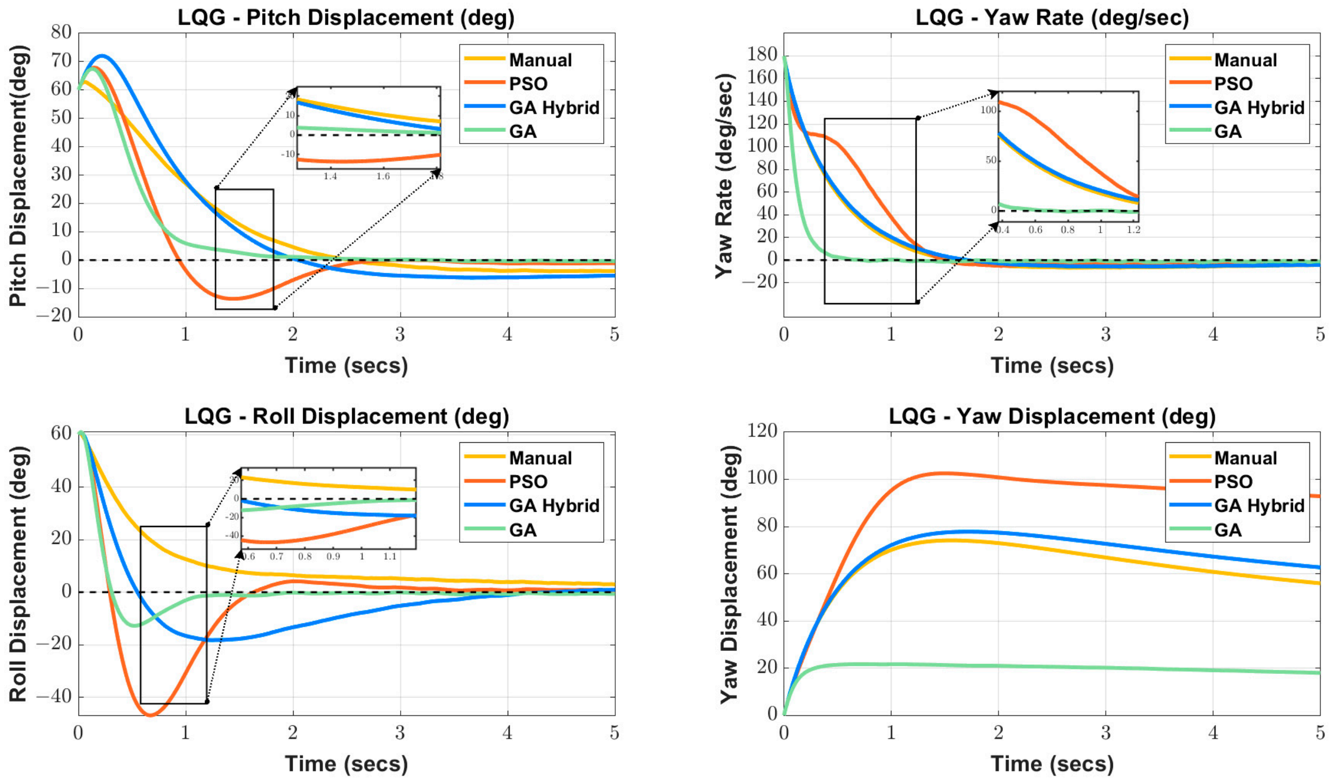


Figure 13. Linear Quadratic Gaussian response for Manually tuned, PSO and GA.

The response for each tuning method for Pitch displacement is very similar with no obvious best method. When compared to roll displacement, both PSO and manually tuned responses possess similar behavior. However, the controller tuned by GA has a larger overshoot although it does become stable faster than other tuning methods. When analyzing yaw rate, the response is in terms of angular velocity, making it hard to compare with the pitch and roll displacement due to the change in unit. To better understand the displacement of the vehicle in its yaw direction the integration of yaw velocity, yaw displacement offers a better insight. Clearly, the controller tuned by GA generates the smallest amount of displacement in the yaw direction only being displaced by 1.7 degrees. The second-best tuning method by process of manual tuning being displaced by a total 26 degrees and the worst preforming controller was tuned by PSO reaching a maximum displacement of 68 degrees.

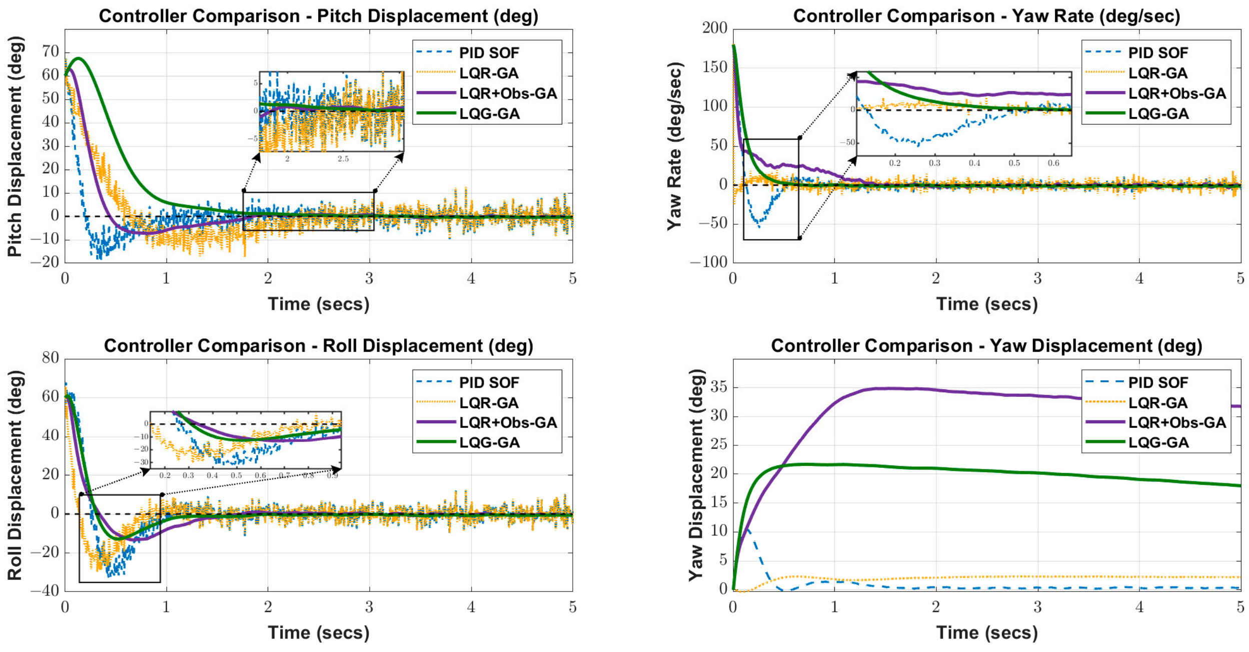


Figure 14. Comparison of the best tuning methods for each controller type.

The response for each of the Full State Compensator tuning methods has better noise attenuation than the Full State Feedback controller due to its state observer (Luenberger observer) to predict the output of the states instead of measuring the noisy outputs directly. However, it is still apparent that there is still some noise in the system such as PSO in Yaw Rate resulting in a particularly poor response. This type of disturbance must be better attenuated if an acceptable controller is to be achieved.

When analyzing pitch displacement both controllers tuned by GA and PSO appear to settle faster than other tuning methods. However, when %OS is considered, the response tuned by PSO overshoots its target by over double the controller tuned by GA. Both the controller tuned by GA-Hybrid and by manual tuning appear to have a similar response both overshooting their target the most out of any tuning method, whilst also taking the longest time to settle.

When analyzing roll displacement, the controller tuned by GA performs the best in both settling time and %OS. The second-best response is the controller tuned by PSO, the %OS is less than that of both manually tuned and GA-Hybrid. The controller tuned by PSO has a settling time similar to GA-Hybrid and the manually tuned controller but not as fast as GA tuned controller. Both manually tuned and GA-Hybrid controller have a similar response in terms of %OS, but GA-Hybrid outperforms manually tuned in terms of settling time.

When analyzing yaw rate, the GA controller falls quickly initially, but flattens slightly at around 50 degrees per second and settles slowly to zero in 1.5 seconds. When this response is analyzed in terms of displacement, the reduction of gradient decent in velocity causes a greater displacement as the controller tuned with GA reaches its highest displacement of 35 degrees. The manually tuned controller shows the best settling time which results in the least maximum displacement.

5.2. Linear Quadratic Gaussian

The responses shown in Figure 13 show the Linear Quadratic Gaussian tuned manually, by PSO, GA and a GA-Hybrid method. When analyzing pitch displacement, the controller tuned by GA stands out as the best in terms of percentage overshoot and on settling time. The controller tuned by PSO also produced an impressive response with slight overshoot, but a faster settling time than manually tuned and GA Hybrid. Manually tuned and GA-Hybrid both fall slowly until zero but overshoot slightly, taking a long time to correct the overshoot back to equilibrium. This is the exact reason that a PSO-equipped controller will outperform manual tuned controller, as demonstrated in Figure 13.

When analyzing roll displacement, it is apparent that the controller tuned by GA has the shortest settling time and the least %OS, which is corrected quickly. The PSO controller greatly overshoots its target value and overshoots on its return to zero, making it an unsuitable controller type. The GA-Hybrid controller has a slight overshoot but a long settling time. Similarly, the controller tuned manually has no overshoot but like GA-Hybrid, the settling time is too long to be considered as an effective control strategy.

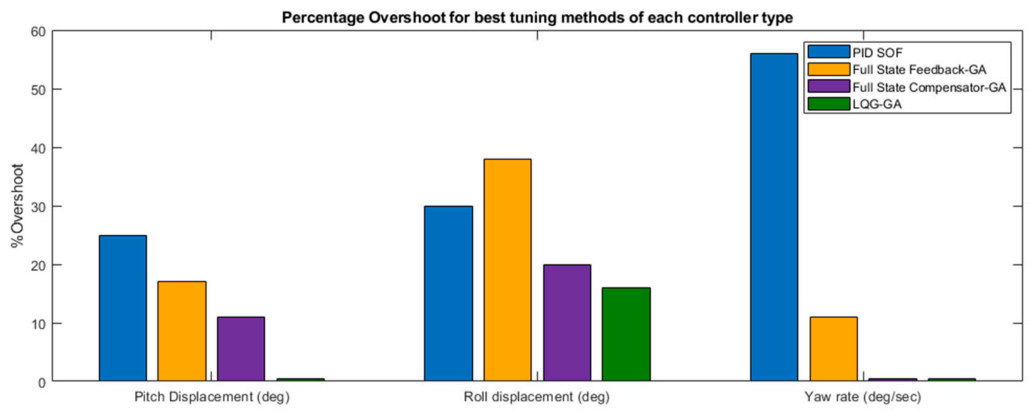
Analysis of yaw displacement and yaw rate shows the controller tuned by the GA method as the standout controller with no overshoot and a very small settling time, the final yaw displacement settling at 21 degrees. There is a slight amount of disturbance present in the yaw rate plot, this is due to the trade off between speed of response and rejection of noise discussed in Section 3.3.3, however, this level of disturbance is an acceptable trade-off for the speed of response. The other tuning methods also settle quickly but overshoot the point of stability slightly and do not stabilize quicky, as they maintain a slight negative velocity meaning these controllers are in a constant negative displacement.

5.3. Comparisons of Best Performing Tuning Method from Each Control Type

The plot in Figure 14 shows the comparison of each of the best tuning methods for each of the different controller types. The benchmark controller "PID SOF" (PID with static output feedback) was included to compare the optimal controllers against the basic conventional method. Having discovered in Sections 5.1 and 5.2 that GA performed best at correcting vehicle offset from wind than any other tuning method, this section aims to compare between controller architectures to find the best architecture for RUAV control.

Firstly, while comparing between controller architectures, it is clear that those controllers that utilize a filter or observer have superior noise attenuation than those without. This is of particular importance in aerial vehicle applications where rotor dynamics can cause large actuator noise, preventing effective correction of the vehicle. Both Full State Feedback tuned with GA (LQR-GA) and PID with Static Output Feedback (PID SOF) are very noisy and produce a response that would be unusable for real applications. However, the controllers that do make use of an observer or filter have better noise attenuation enabling a clean signal to be preserved, enabling it to be used in real applications.

Figure 15 can be used to easily evaluate the performance of each controller type in percentage overshoot and settling time. When evaluating controllers on settling time, the GA tuned LQG (LQG-GA) controller outperforms every other controller tested in both roll displacement and yaw rate. Similarly, when analyzing percentage overshoot performance, LQG-GA is clearly the best controller of those tested as it has the least overshoot for parameters of pitch displacement, roll displacement and yaw rate. This ensures that the vehicle is moved from a place of state offset back to a constant hover state with no overshoot of its target providing good vehicle response for correction under harsh conditions. PID-SOF does have a better settling time than LQG for pitch displacement, but due to its poor attenuation of noise, this controller is not a viable option to consider as the best controller. It can be concluded from this section the LQG is the best controller of those tested for RUAV state correction in cases of high external disturbance due to wind and sensor noise.



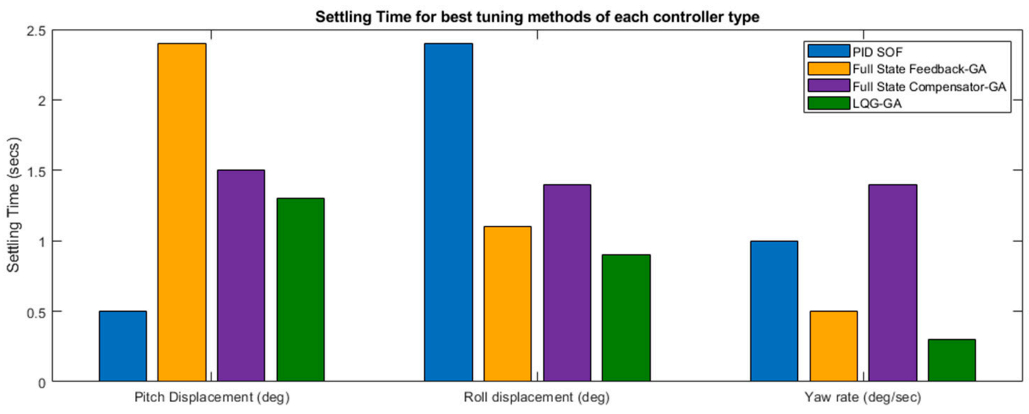


Figure 15. Percentage Overshoot and settling time for best controllers.

6. Future Work

As discussed in the previous section, Linear Quadratic Gaussian controllers tuned using machine learning methods such as Genetic Algorithm are capable of outperforming those same controllers tuned by manually. This poses a new and exciting question:

“Can an optimal controller be replaced by a more advanced machine model such as Reinforcement Learning to learn a specific control law completely from scratch?”

Reinforcement Learning

Reinforcement Learning (RL) is a fast-growing area of machine learning that enables artificially intelligent agents to react to changes in an environment with the motivation of accumulating reward.

Figure 16 presents the block diagram for the Reinforcement Learning process. The Agent makes decisions based on the policy, this policy is updated by a reinforcement learning algorithm based on rewards obtained for the previous action. Over time, the agent will learn to act in a way that maximizes its rewards from the environment. By introducing an agent to control the RUAV the controller is no longer required to define the conventional control laws as the agent learns them itself.

However, by using a controller such as Linear Quadratic Gaussian over Reinforcement Learning, the implemented controller has no complexity and, therefore, no large computational overheads. In order to make reinforcement learning a viable option for machine learning control, the complexity of such models must be reduced to enable lightweight controllers to be created for real applications.

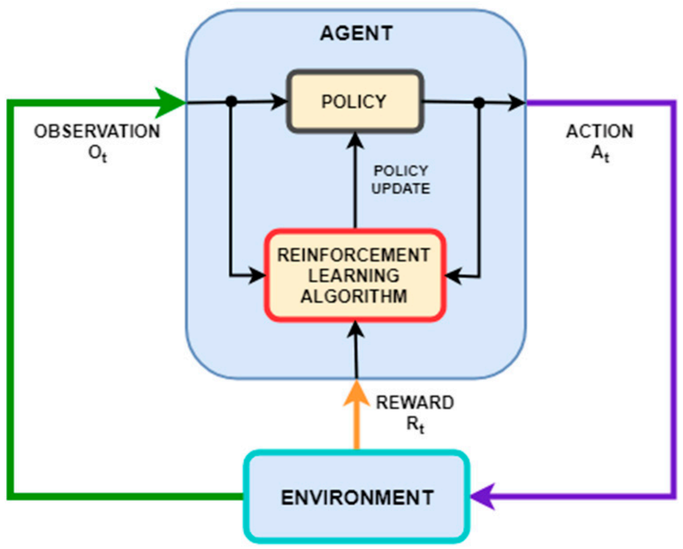


Figure 16. Reinforcement Learning Block Diagram [26].

7. Conclusions

This work has demonstrated that the transient response of optimal controllers can be improved by tuning using machine learning metaheuristic algorithms. Firstly, the conventional method of manual tuning was tested on well-known state-of-the-art methods such as Full State Feedback, Full State Compensator and Linear Quadratic Gaussian controllers. Secondly, metaheuristic algorithms (Particle Swam Optimization and Genetic Algorithm) were evaluated for their tuning capabilities for the same controller architectures. This demonstrated that improvements in performance are possible by choosing an algorithmic approach to tuning controller gains. The best controller design tested was the Linear Quadratic Gaussian tuned by Genetic Algorithm, this controller outperformed the same controller tuned by manual tuning, Particle Swarm Optimization, and a Genetic Algorithm Hybrid approach. The best controller (LQG-GA) had the fastest settling time outperforming manual tuning, Particle Swarm Optimization and GA-Hybrid by 20%, 17% and 21%, respectively, while attenuating measurement and process noise present in the system. There is a high computational cost associated with running optimization algorithms for tuning controller gains but as this is completed offline, the computational expense to the RUAV in flight remains the same. The key to successful implementation of PSO or GA is in the selection of an effective Fitness Function. Currently, there are still shortcomings associated with zero convergence criteria but this improvement in transient response induced by machine learning is a key step towards implementing safe, year-round flight of RUAVs in Northern Ireland.

Author Contributions: Conceptualization, J.G.; methodology, J.G.; software, J.G.; validation, J.G.; formal analysis, J.G.; investigation, J.G.; resources, J.G.; data curation, J.G.; writing—original draft preparation, J.G.; writing—review and editing, J.G. and M.U.H.; visualization, J.G.; supervision, M.U.H.; project administration, J.G. and M.U.H.; funding acquisition, M.U.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

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

Appendix A. State Space Model

Matrix Name	Values
State Matrix (A)	$\begin{bmatrix} -0.019 & 0.017 & 0.384 & -9.792 & -0.001 & -0.337 & 0.000 & 0.000 \\ 0.014 & -0.299 & 0.024 & -0.586 & -0.002 & -0.026 & 0.537 & 0.000 \\ 0.041 & -0.003 & -1.839 & 0.000 & 0.002 & 0.528 & 0.000 & -0.002 \\ 0.000 & 0.000 & 0.999 & 0.000 & 0.000 & 0.000 & 0.000 & 0.055 \\ 0.001 & -0.002 & -0.338 & 0.032 & -0.035 & -0.403 & 9.778 & 0.117 \\ 0.013 & 0.000 & -3.047 & 0.000 & -0.229 & -10.620 & 0.000 & -0.033 \\ 0.000 & 0.000 & -0.003 & 0.000 & 0.000 & 1.000 & 0.000 & 0.060 \\ 0.002 & 0.006 & -0.541 & 0.000 & 0.004 & -1.855 & 0.000 & -0.349 \end{bmatrix}$
Input Matrix (B)	$\begin{bmatrix} -10.350 & 1.079 & 0.000 \\ -0.729 & 0.076 & 0.000 \\ 27.090 & -4.724 & -0.186 \\ 0.000 & 0.000 & 0.000 \\ -1.082 & -10.370 & 4.724 \\ -27.290 & -156.400 & -1.069 \\ 0.000 & 0.000 & 0.000 \\ -4.897 & -27.970 & -12.930 \end{bmatrix}$
Output Matrix—Original (C)	$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$
Output Matrix—Simplified (C)	$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$
Feed Forward Matrix—Original (D)	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
Feed Forward Matrix—Simplified (D)	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

Appendix B. Simulation Parameters for Full State Feedback and PID Controller

Model Parameters	Values	
Initial Conditions applied to state matrix	$[2; 2; 90; 60; 2; 60; 60; 180]$	
Measurement (Sensor) Noise	Gaussian Noise	Noise power: 0.05 Sample frequency: 200 Hz
Process (Actuator) Noise	Random Number	Mean: 0 Variance: 0.5 Sampling frequency: 10 Hz
Parameters for Manually Tuned	Values	
Q Matrix—Manually Tuned	$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 30 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 9 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 20 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 20 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix}.$	

Model Parameters	**Values**
R Matrix—Manually Tuned	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 10 \end{bmatrix}$
Full State Feedback Gain (K)—Manually Tuned	$\begin{bmatrix} -0.950 & -1.325 & 2.619 & 7.073 & -0.315 & -0.715 & -2.555 & -0.157 \\ 0.183 & -0.069 & -0.402 & -1.405 & -1.902 & -1.902 & -12.789 & -1.030 \\ 0.011 & 0.004 & -0.002 & -0.109 & 0.160 & 0.151 & 0.200 & -0.935 \end{bmatrix}$
Parameters for PSO algorithm	Values
N_particles	11
Swarm Size	200
c1	2.4
c2	2.44
MaxIter	100
Wmin	[0, 0, 0, 0, 0, 0, 0, 0, 0.001, 0.001, 0.001]
Wmax	[1000, 1000, 1000, 1000, 1000, 1000, 1000, 1000, 0.01, 0.01, 0.01]
Full State Feedback Gain (K)—PSO	$\begin{bmatrix} -171.9 & -36.8 & 242.8 & 880.8 & -34.9 & -54.8 & -237 & -16.6 \\ 31.7 & -5.8 & -35.6 & -179.6 & -251.1 & -314.8 & -1235.7 & -100.5 \\ 12.2 & -0.9 & 0 & -88.2 & 149.5 & 13.1 & -81.2 & -144.4 \end{bmatrix}$
Parameters for GA	Values
PopNum	100
ChromNum	11
CrossVal	80
MutVal	20
MaxGen	200
Full State Feedback Gain (K)—GA	$\begin{bmatrix} -5.367 & -1.521 & 10.926 & 34.420 & -1.558 & -0.223 & -3.954 & -1.408 \\ 0.999 & -0.083 & -1.607 & -6.615 & -9.297 & -1.420 & -18.133 & -7.068 \\ 0.736 & 0.342 & -0.405 & -5.938 & 20.098 & 1.375 & 16.020 & -31.802 \end{bmatrix}$
Parameters for PID with SOF	Values

Pitch displacement PID controller gains	Kp = −4.887
	Ki = 13.427
	Kd = 81.438

Roll displacement PID controller gains	Kp = 0.199
	Ki = −2.622
	Kd = −53.927

Roll displacement PID controller gains	Kp = 1.900
	Ki = −2.010
	Kd = −241.273

Static Output Feedback gain (SOF)	$\begin{bmatrix} 6.643 & 0.360 & 0.251 & 0.702 & 0.006 \\ -0.201 & -1.027 & 0.072 & -0.026 & -0.075 \\ 0.205 & 0.240 & -1.189 & -0.026 & 0.157 \end{bmatrix}$

Appendix C. Simulation Parameters for Full State Compensator

Model Parameters	**Values**	
Initial Conditions applied to state matrix	$[2; 2; 90; 60; 2; 60; 60; 180]$	
Measurement (Sensor) Noise	Gaussian Noise	Noise power: 0.05 Sample frequency: 200 Hz
Process (Actuator) Noise	Random Number	Mean: 0 Variance: 0.5 Sampling frequency: 10 Hz
Parameters for Manually Tuned	Values	

Model Parameters	**Values**
Luenberger Q Matrix—Manually Tuned	$B * B^T$
Luenberger R Matrix—Manually Tuned	$\begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{bmatrix}$
Luenberger Observer Gain (Lo)—Manually Tuned	$\begin{bmatrix} -14.308 & -1.816 & 0.944 \\ -1.055 & 0.379 & 0.069 \\ 25.547 & 2.971 & -1.546 \\ 7.138 & 0.045 & -0.206 \\ -0.330 & 13.422 & 7.888 \\ -2.164 & 60.376 & 132.850 \\ 0.0453 & 10.981 & 0.439 \\ -0.206 & 0.439 & 37.805 \end{bmatrix}$
Parameters for PSO algorithm	Values
N_particles	22
Swarm Size	20
c1	1
c2	1
MaxIter	2000
Wmin	[0, 0, 0, 0, 0, 0, 0, 0, 0.001, 0.001, 0.001, 0, 0, 0, 0, 0, 0, 0, 0, 0.001, 0.001, 0.001]
Wmax	[1000, 1000, 1000, 1000, 1000, 1000, 1000, 1000, 0.01, 0.01, 0.01,1000, 1000, 1000, 1000, 1000, 1000, 1000, 1000, 0.01, 0.01, 0.01]
State Feedback Gain (K)—PSO	$\begin{bmatrix} -370 & -13.1 & 191.9 & 1097.4 & -50.3 & -32.1 & -212 & -28.6 \\ 70.1 & -3.7 & -23.5 & -214.8 & -299.8 & -195.4 & -1051.5 & -152.2 \\ 12.6 & 0 & -0.2 & -76 & 181 & 26.8 & 94.5 & -243.8 \end{bmatrix}$
Luenberger Observer Gain (Lo)—PSO	$\begin{bmatrix} 55.987 & -21.564 & 1.663 \\ 2.671 & 0.320 & 1.789 \\ 53.243 & -14.659 & -6.605 \\ 255.968 & -0.058 & 0.005 \\ -27.694 & -125.848 & 190.491 \\ -11.456 & 30.085 & -31.376 \\ -0.058 & 198.186 & -0.120 \\ 0.005 & -0.120 & 310.408 \end{bmatrix}$
Parameters for GA-Hybrid	**Values**
PopNum	100
ChromNum	9
CrossVal	80
MutVal	60
MaxGen	2000
State Feedback Gain (K)–GA-Hybrid	$\begin{bmatrix} -0.458 & -0.064 & 1.049 & 3.095 & -0.092 & -0.142 & -0.658 & -0.016 \\ 0.108 & -0.014 & -0.181 & -0.882 & -1.364 & -1.575 & -6.282 & -0.468 \\ 0.032 & -0.003 & -0.020 & -0.331 & 0.707 & 0.003 & 0.214 & -0.297 \end{bmatrix}$
Parameters for GA	Values
PopNum	50
ChromNum	18
CrossVal	80
MutVal	60
MaxGen	5000
State Feedback Gain (K)—PSO	$\begin{bmatrix} -0.718 & -0.112 & 0.776 & 3.320 & -0.131 & -0.080 & -0.637 & -0.041 \\ 0.155 & -0.029 & -0.116 & -0.731 & -0.671 & -0.485 & -2.766 & -0.282 \\ 0.027 & -0.006 & -0.010 & -0.241 & 0.309 & 0.070 & 0.321 & -0.538 \end{bmatrix}$

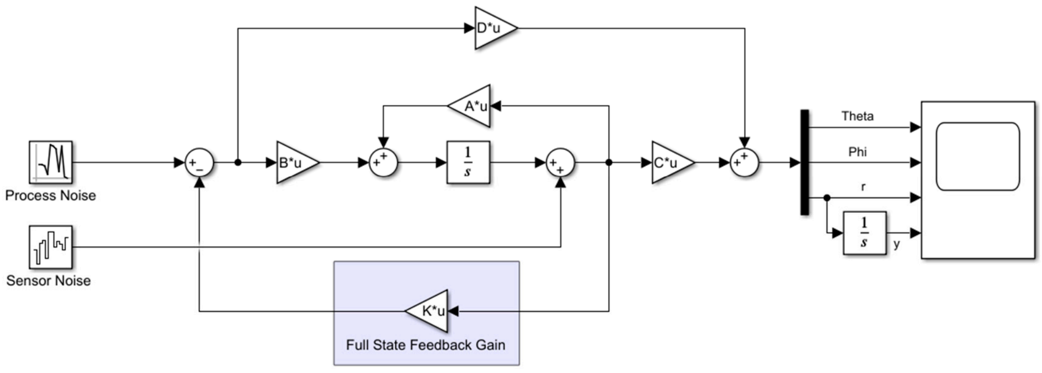
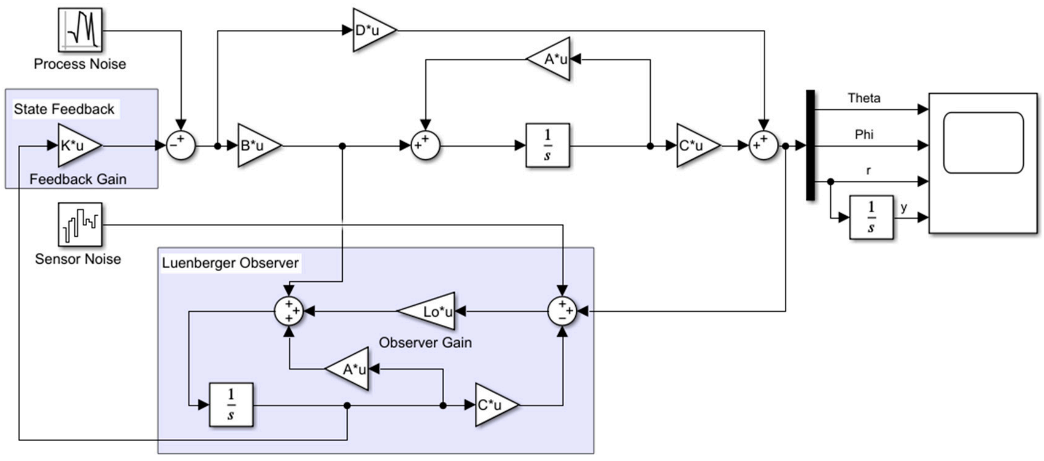
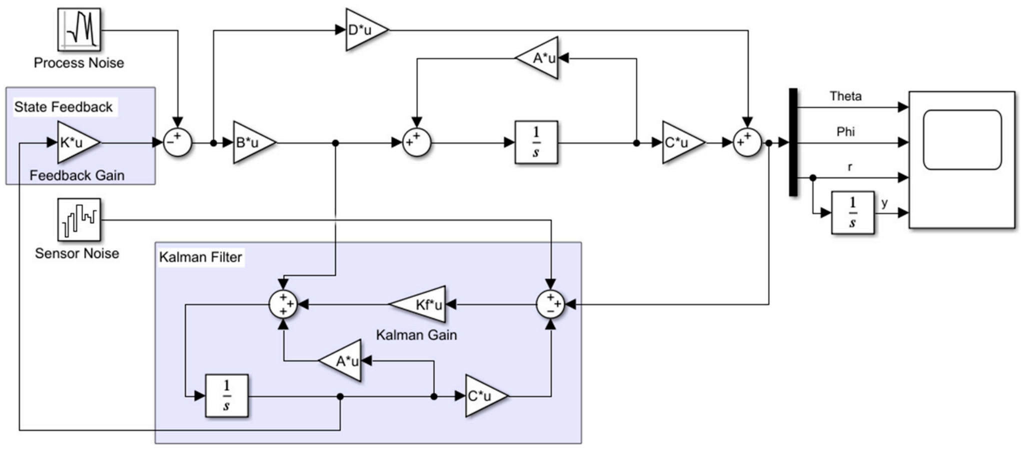
Model Parameters	Values
Luenberger Observer Gain (Lo)—GA	$\begin{bmatrix} 8.040 & -7.643 & 1.492 \\ 0.263 & -0.044 & 0.118 \\ 69.133 & 5.712 & -5.461 \\ 11.747 & -0.120 & -0.188 \\ -3.695 & 0.349 & 2.426 \\ -8.400 & 255.689 & -134.315 \\ -0.120 & 22.230 & -4.739 \\ -0.188 & -4.739 & 105.187 \end{bmatrix}$

Appendix D. Simulation Parameters for Linear Quadratic Gaussian

Model Parameters	Values	
Initial Conditions applied to state matrix	[2; 2; 90; 60; 2; 60; 60; 180]	
Measurement (Sensor) Noise	Gaussian Noise	Noise power: 0.05 Sample frequency: 200 Hz
Process (Actuator) Noise	Random Number	Mean: 0 Variance: 0.5 Sampling frequency: 10 Hz
Parameters for Manually Tuned	Values	
Feedback gain Q Matrix—Manually Tuned	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 50 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 200 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 50 \end{bmatrix}$	
Feedback gain R Matrix—Manually Tuned	$\begin{bmatrix} 0.01 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.01 \end{bmatrix}$	
State Feedback Gain (K)—Manually Tuned	$\begin{bmatrix} 0.001 & 0 & 2.170 & 69.595 & 0 & -0.154 & -13.638 & 10.375 \\ 0 & 0 & -0.364 & -12.507 & 0.001 & -0.911 & -74.072 & -59.003 \\ 0 & 0 & 0 & 0.189 & -0.003 & 1.826 & 119.697 & -37.521 \end{bmatrix}$	
Vd Matrix—Manually Tuned	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.06 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.06 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.06 \end{bmatrix}$	
Vn Matrix—Manually Tuned	$\begin{bmatrix} 0.2 & 0 & 0 \\ 0 & 0.2 & 0 \\ 0 & 0 & 0.2 \end{bmatrix}$	
Kalman Gain (Kf)—Manually Tuned	$\begin{bmatrix} 0.959 & -1.887 & -0.289 \\ -0.133 & 0.273 & 0.145 \\ 0.078 & -0.042 & -0.042 \\ 1.776 & -0.014 & 0.005 \\ -0.752 & 5.649 & 7.882 \\ -0.010 & -0.084 & -0.162 \\ -0.014 & 1.684 & 0.037 \\ 0.005 & 0.037 & 1.608 \end{bmatrix}$	

Model Parameters	**Values**
Parameters for PSO algorithm	Values
N_particles	18
Swarm Size	20
c1	1
c2	1
MaxIter	2000
Wmin	[0, 0, 0, 0, 0, 0, 0, 0, 0.001, 0, 0, 0, 0, 0, 0, 0, 0, 0.001]
Wmax	[1000, 1000, 1000, 1000, 1000, 1000, 1000, 1000, 0.01,1000, 1000, 1000, 1000, 1000, 1000, 1000, 1000, 0.01]
Full State Feedback Gain (K)—PSO	$\begin{bmatrix} -0.655 & -0.187 & 0.690 & 3.177 & -0.014 & -0.064 & -0.341 & -0.014 \\ 0.170 & -0.292 & -0.120 & -0.696 & -0.058 & -0.393 & -1.342 & -0.0252 \\ 0.010 & 0.065 & -0.002 & -0.141 & 0.012 & 0.139 & 0.315 & -0.902 \end{bmatrix}$
Kalman Gain (Kf)—PSO	$\begin{bmatrix} -1.564 & -1.223 & 0.040 \\ -0.171 & 0.415 & 0.079 \\ 0.102 & -0.019 & -0.031 \\ 3.683 & -0.004 & 0.023 \\ -0.804 & 6.969 & 6.889 \\ -0.016 & -0.072 & -0.149 \\ -0.004 & 5.847 & 0.028 \\ 0.023 & 0.028 & 4.884 \end{bmatrix}$
Parameters for GA-Hybrid	**Values**
PopNum	100
ChromNum	9
CrossVal	80
MutVal	20
MaxGen	5000
State Feedback Gain (K)—GA-Hybrid	$\begin{bmatrix} -0.458 & -0.064 & 1.049 & 3.095 & -0.092 & -0.142 & -0.658 & -0.016 \\ 0.108 & -0.014 & -0.181 & -0.882 & -1.364 & -1.575 & -6.282 & -0.468 \\ 0.032 & -0.003 & -0.020 & -0.331 & 0.707 & 0.003 & 0.214 & -0.297 \end{bmatrix}$
Parameters for GA	Values
PopNum	1000
ChromNum	18
CrossVal	80
MutVal	20
MaxGen	50
State Feedback Gain (K)—GA	$\begin{bmatrix} 0 & 0 & 0.220 & 1.071 & -0.001 & 0 & -0.133 & -0.166 \\ 0 & 0 & -0.041 & -0.292 & -0.001 & -0.029 & -0.440 & -0.790 \\ 0 & 0 & 0.011 & 0.036 & -0.001 & 0.105 & 0.371 & -0.953 \end{bmatrix}$
Kalman Gain (Kf)—GA	$\begin{bmatrix} -3.401 & -1.289 & -0.453 \\ -0.266 & 0.435 & 0.015 \\ 0.100 & -0.018 & -0.035 \\ 9.496 & -0.002 & 0.024 \\ -0.730 & 7.143 & 5.313 \\ -0.016 & -0.053 & -0.120 \\ -0.002 & 9.248 & 0.031 \\ 0.024 & 0.031 & 9.982 \end{bmatrix}$

Appendix E. Controller Architectures

Controller Name	Block Diagram
Full State Feedback	
Full State Compensator	
Linear Quadratic Gaussian	

References

1. Met Eireann. Storm Eunice to Bring Strong Winds Rain and Snow. 2022. Available online: https://www.met.ie/storm-eunice-to-bring-strong-winds-rain-and-snow (accessed on 18 April 2022).
2. DJI. DJI MATRICE 300 RTK—User Manual. 2020. Available online: https://dl.djicdn.com/downloads/matrice-300/20200507/M300RTKUserManualEN.pdf (accessed on 18 April 2022).

3. Zhang, Z. Application of PID Simulation Control Mode in Quadrotor Aircraft. In Proceedings of the 2020 International Conference on Computer Engineering and Application, Guangzhou, China, 18–20 March 2020.
4. Lahlouh, I.; Rerhrhaye, F.; Elakkary, A.; Sefiani, N.; Bybi, A. A combined static output feedback-PID control for TITO process based particle swarm optimization: Simulation and practical implementation for the poultry house system. *Int. J. Dyn. Control* **2022**, 1–14. [CrossRef]
5. Pandey, S.K.; Laxmi, V. Optimal Control of Twin Rotor MIMO System Using LQR Technique. In *Computational Intelligence in Data Mining*; Springer: New Delhi, India, 2015; Volume 1, pp. 11–21.
6. De-Xin, G.; Bao-tong, C. LQR Controller Design of MIMO Systems with External disturbances based on Stability Degree Constraint. In Proceedings of the 2010 IEEE International Conference on Mechatronics and Automation, Xi'an, China, 4–7 August 2010.
7. Chen, B.; Yu, C.; Hsu, W. Steering control of six-wheeled vehicles using linear quadratic regulator techniques. *J. Automob. Eng.* **2007**, *221*, 1231–1240. [CrossRef]
8. Barzanooni, E.; Salahshoor, S.; Khaki-Sedigh, A. Attitude flight control system design of UAV using LQG\LTR multivariable control with noise and disturbance. In Proceedings of the 2015 3rd RSI International Conference on Robotics and Mechatronics (ICROM), Tehran, Iran, 7–9 October 2015.
9. Panponpen, K.; Konghirur, M. LQR state feedback controller based on particle swarm optimization for IPMSM drive system. In Proceedings of the IEEE 10th Conference on Industrial Electronics and Applications (ICIEA), Auckland, New Zealand, 15–17 June 2015.
10. Bartyś, M.; Hryniewicki, B. The Trade-Off between the Controller Effort and Control Quality on Example of an Electro-Pneumatic Final Control Element. *Actuators* **2019**, *8*, 23. [CrossRef]
11. Yu, G.; Hsieh, P. Optimal Design of Helicopter Control Systems Using Particle Swarm Optimization. In Proceedings of the IEEE International Conference on Industrial Cyber Physical Systems (ICPS), Taipei, Taiwan, 6–9 May 2019.
12. Walker, D.J.; Postlethwaite, I. Advanced helicopter flight control using two-degree-of-freedom H(infinity) optimization. *J. Guid. Control. Dyn.* **1996**, *19*, 461–468. [CrossRef]
13. Boukhnifer, M.; Chaibet, A.; Larouci, C. H-infinity robust control of 3-DOF helicopter. In Proceedings of the International Multi-Conference on Systems, Signals & Devices, Chemnitz, Germany, 20–23 March 2012.
14. Huang, C.; Zhang, H. Comparison of Disturbance Rejection Performance between Three Types of UAV Linear Controllers. In Proceedings of the 7th International Conference on Information Science and Control Engineering (ICISCE), Changsha, China, 18–20 December 2020.
15. Lee, K.; Kim, S.; Kwak, S.; You, K. Quadrotor Stabilization and Tracking Using Nonlinear Surface Sliding Mode Control and Observer. *Appl. Sci.* **2021**, *11*, 1417. [CrossRef]
16. Madani, T.; Benallegue, A. Backstepping Control for a Quadrotor Helicopter. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China, 9–15 October 2006.
17. Hwangbo, J.; Sa, I.; Siegwart, R.; Hutter, M. Control of a Quadrotor with Reinforcement Learning. *IEEE Robot. Autom. Lett.* **2017**, *2*, 2096–2103. [CrossRef]
18. Waslander, S.L.; Hoffmann, G.M.; Jang, J.S.; Tomlin, C.J. Multi-agent quadrotor testbed control design: Integral sliding mode vs. reinforcement learning. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China, 9–15 October 2006.
19. Le, T.; Quynh, N.V.; Long, N.K.; Hong, S. Multilayer Interval Type-2 Fuzzy Controller Design for Quadcopter Unmanned Aerial Vehicles Using Jaya Algorithm. *IEEE Access* **2020**, *8*, 181246–181257. [CrossRef]
20. Al-Sharmn, M.K.; Zweiri, Y.; Jaradat, M.A.K.; Al-Husari, R.; Gan, D.; Seneviratne, L.D. Deep-Learning-Based Neural Network Training for State Estimation Enhancement: Application to Attitude Estimation. *IEEE Trans. Instrum. Meas.* **2019**, *69*, 24–34. [CrossRef]
21. Hadi, M.U. Practical Demonstration of 5G NR Transport Over-Fiber System with Convolutional Neural Network. *Telecom* **2022**, *3*, 103–117. [CrossRef]
22. Brunton, S.L.; Kutz, J.N. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems and Control*; Cambridge University Press: Cambridge, UK, 2019.
23. Choubey, C.; Ohri, J. Parallel Manipulator control using different LQG tuning methods. In Proceedings of the 2018 5th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON), Gorakhpur, India, 2–4 November 2018.
24. MathWorks. Multiloop Control of a Helicopter. 2020. Available online: https://uk.mathworks.com/help/control/ug/multi-loop-control-of-a-helicopter.html (accessed on 18 April 2022).
25. Hadi, M.U.; Jung, H.; Traverso, P.A.; Tartarini, G. Experimental evaluation of real-time sigma-delta radio over fiber system for fronthaul applications. *Int. J. Microw. Wirel. Technol.* **2020**, *13*, 756–765. [CrossRef]
26. MathWorks. Reinforcement Learning Onramp. 2021. Available online: https://uk.mathworks.com/learn/tutorials/reinforcement-learning-onramp.html (accessed on 18 April 2022).