

# Intelligent PID Controller Based on Neural Network for AI-Driven Control Quadcopter UAV

Nur Hayati Sahrir <sup>a,1,\*</sup>, Mohd Ariffanan Mohd Basri <sup>a,2,\*</sup>

<sup>a</sup> Faculty of Electrical Engineering, Universiti Teknologi Malaysia, 81310 Johor Bahru, Johor, Malaysia

<sup>1</sup> [nurhayatisahrir1998@graduate.utm.my](mailto:nurhayatisahrir1998@graduate.utm.my); <sup>2</sup> [ariffanan@fke.utm.my](mailto:ariffanan@fke.utm.my)

\* Corresponding Author

## ARTICLE INFO

### Article history

Received March 27, 2024

Revised May 05, 2024

Accepted May 14, 2024

### Keywords

Feedforward Neural Network;  
Particle Swarm Optimization;  
PID Controller;  
Quadcopter UAV

## ABSTRACT

Unmanned Aerial Vehicle (UAV), specifically a quadcopter is publicly popular which it provides services in different applications such as aerial delivery, aerial photography, military, weather forecasting and more examples to date. A Proportional-Integral-Derivative (PID) controller is one of the control techniques that can provide stabilization and reliable trajectory tracking. However, proper PID gains are needed to ensure a stable flight and it should be hybridized or improved to increase the robustness, reliability, and stabilization during flight. In this paper, an intelligent PID controller using neural network is proposed based on Levenberg-Marquardt feedforward neural network training method. The PID gains are initialized using different ranges according to the optimal gains generated by Particle Swarm Optimization, and this contributes towards a good training performance using Mean Square Error (MSE) evaluation. The trained network takes desired output and references as input data to calculate the required combination of PID gains as the output. The including of the response characteristics as the input data for the network, together with reference, error, and control input is the significance of the work. The performance of this work is presented using MSE performances, attitudes and altitude stabilization, and trajectory tracking reliability through error index performances. The simulation results graphically prove that the proposed controller provides better stability with reduced overshoot and settling times. Disturbance rejection is also enhanced by 1.7% compared to manual tuned PID controller. The reliability of the proposed controller highlights avenues for further exploration in AI-driven control strategies for quadcopter systems.

This is an open-access article under the [CC-BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



## 1. Introduction

It is widely known that unmanned aerial vehicles (UAV) provide many useful applications in various fields such as aerial delivery [1], aerial photography [2], military [3], weather forecasting [4] and more examples to date. Although drones have been publicly commercialized and utilized, the research and development on control of drones is still in progress to further improve the controller. Quadcopter, in specific, has the benefits of better stabilizing mechanism, less complex and cost-effective, if compared to other multi-rotor drones. However, a quadcopter model alone has challenges to fly properly in the air. Its non-linear characteristics [5] and coupled dynamics [6] make a quadcopter as an underactuated system to be vulnerable towards perturbations without a controller to correct the

state error [7]. With the right controller in the closed-loop system, a quadcopter can have six degrees of freedom with only four rotors. There are several control techniques established in the last few years involving linear, non-linear and learning-based controllers [8] but the Proportional-Integral-Derivative (PID) stands out the most as the simplest controller model.

PID controller is the most common controller in any system with the advantage of its flexibility and straightforward design. It is possible to control a firefighter robot using PID controller [9]. The work [10] uses PID control algorithm in MATLAB/Simulink to control the motor speed effectively in terms of application and operational. Both PID controller and overhead electrical resistance are evaluated to control the temperature in a pig nursery and PID controller performs better in the resistance heating system providing comfort conditions during the coldest hours [11]. Another example to show the usefulness of PID controller is the controller application in an automatic solar-powered lawnmower [12] by using a two-degree of freedom PID controllers. For a quadcopter system instead, the work [13]-[16] used a cascaded PID controller design to handle the underactuated quality of the system. Both simulation and experimental results are satisfying. Both work [17], [18] modifies the classic PID equation into a nonlinear equation by adding some nonlinear functions and better input tracking results of quadcopter are obtained. By using Parrot Mambo quadcopter model, the work [19] proves the efficiency of classical PID controller to control quadcopter flight during both simulation and experiment under small perturbations. These studies highlight the variety of applications and significance of these controller types.

The determination of the PID gains is the key for designing a PID controller. Although the work [20] has proved that the steady state error in the modelling process is an issue that the PID controller can handle well, an appropriate tuning of PID gains is essential for fast dynamic response and high controllability [21]. The typical tuning method of Ziegler Nichol's is less efficient on quadcopter system which improvement is needed as in work [22]. An alternative method to tune PID gains for a quadcopter is by using metaheuristic algorithms such as Particle Swarm Optimization (PSO) [23]-[26] and Genetic Algorithm (GA) [27]-[29]. Both algorithms iteratively search for optimal gains depending on the best objective function. Review and analysis of different other algorithms on engineering applications were made in [30], [21].

While metaheuristic algorithms provide an efficient search for optimal solutions, they lack the ability to deal with highly nonlinear and multimodal objective functions. Neural networks, on the other hand, have the ability to process the complex relationships between system dynamics and PID gains. To develop a successful practical application, the PID controller typically needs sort of a priori manual retuning, and it is challenging to provide disturbance rejection, low oscillation, and faster settling time capabilities for a quadcopter system at the same time. Understanding how to make a quadcopter respond effectively to a particular flight style while considering the various flight characteristics is a crucial thing. This objective can be achieved by using neural network to auto-tune PID gains for quadcopter which satisfying results are proven from previous literatures [31]-[33].

This work aims to develop an auto-tuned PID controller using feedforward neural network (FFNN) to produce a better trajectory tracking performance of quadcopter when compared to manually tuned PID controller. PID has simple structure that can easily be adapted with other controllers or optimization techniques. By referring to the work [34], this work specifically address the method of generating an offline dataset that includes the combinations of quadcopter dynamic responses with respective PID gains, for neural network training to produce a more adaptive PID gains for quadcopter, in an offline manner. In this work, a dataset of five hundred combinations is formed. By using different initial gain ranges obtained from manual tuned PID gains for each quadcopter's dynamic states ( $\phi, \theta, \gamma, z$ ), it helps to narrow down the choices of each gain. Different parameters require a specifically appropriate tuned PID gains, and they produce different results. This idea produces better quadcopter performances if compared to using the same ranges. Moreover, the network training is using inputs including response characteristics, setpoints, states error and control inputs of quadcopter, with PID gains as the calculated outputs. MATLAB/Simulink software is used as the simulation environment to build the quadcopter model and run the whole proposed control

system while comparing the results with the manual tuned PID controller performance. This work contributes to the following aspects:

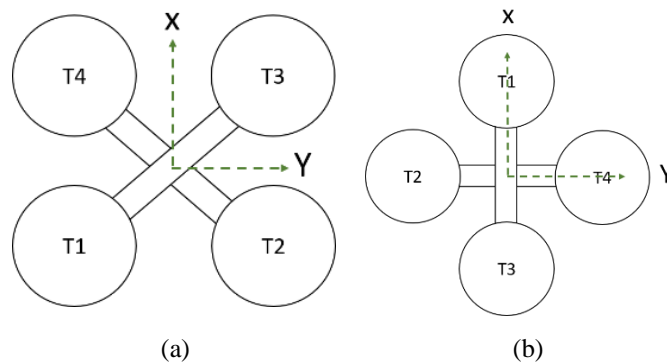
1. It is difficult to analyze the whole PID manual tuning process to obtain good gains. Neural networks can help to analyze the dataset that contains different dynamic responses which correspond to different sets of PID gains.
2. Investigate the reliability of the proposed method by observing improved performance over manual tuned PID. It was found that this method helps for performance improvement.
3. This work is partly ideated from [34]. The research key innovation is the dataset formed using gains from manual tuning process and the different cases of network that determine the performance of quadcopter control.

This paper is structured into four major parts where it started with [Section 1](#) that explains in detail the state-of-the-art, a brief of this work and its findings. [Section 2](#) introduces the quadcopter details on both the physical dynamics and modelling where the formulation is based on Newton-Euler. It also includes the controller design methodology that is based on PID controller and neural network training with offline dataset generation with the selection of the best network. The results of the network testing, altitude and attitudes stabilization and position tracking are all included in [Section 3](#) and [Section 4](#) concludes this work by relating the objectives, results, findings, and recommendations for future work to provide a clear summary of this work. While this work focuses on quadcopter flight analysis using neural network to produce optimal PID gains, future research directions will focus on using neural networks for real-time experiment to further advance our understanding of underlying reasons for specific parameter adjustments.

## 2. Research Methodology

### 2.1. Quadcopter Modelling

According to [Fig. 1](#), there are two different quadcopter configurations: the “cross” configuration and the “plus” configuration. Most studies employ a “plus” configuration, which ignores the quadcopter's nonlinear effect and significantly increased agility [23]. However, because it uses two rotors to act during any movement, a “cross” design is thought to be more stable [35].

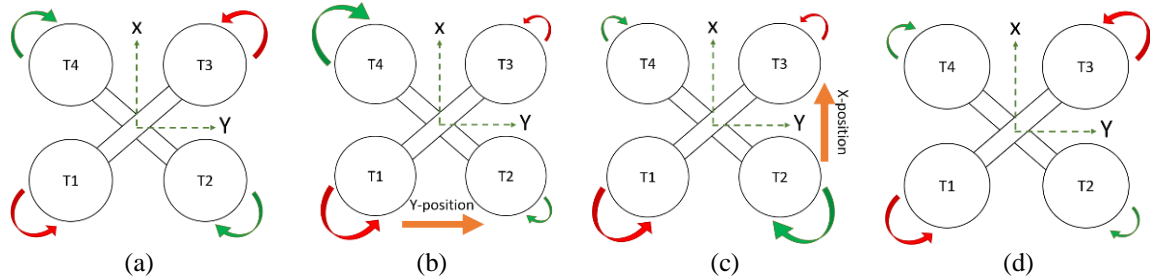


**Fig. 1.** (a) 'Cross' (b) 'Plus'

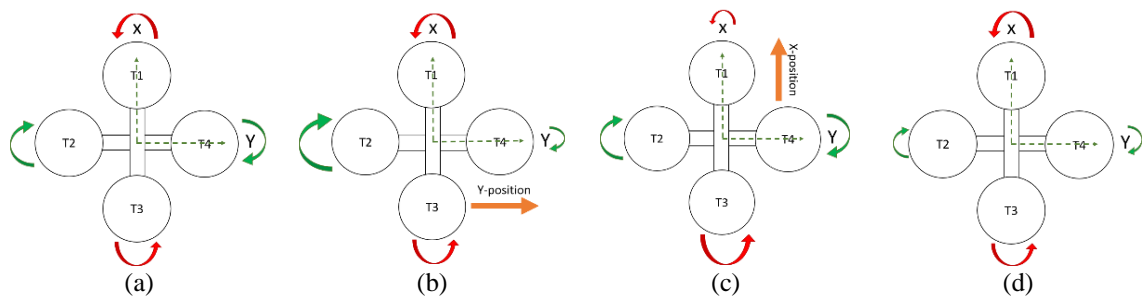
For instance, in a “cross” layout, the speed of rotors 1 and 2 (3 and 4) will increase (decrease) simultaneously during pitch movement. However, only rotor 1 (rotor 3) speed will decrease (increase) in the “plus” configuration. Since quadcopter’s motions are in effect with the Newton’s Third Law concept where, for every action, there is an equal and opposite reaction, the movements that a quadcopter can make by varying the speeds of each rotor are depicted in [Fig. 2](#) and [Fig. 3](#) [36].

The quadcopter may be lifted vertically to a specific altitude and can hover in the air due to the four identically speeded rotors as shown in [Fig. 2](#) (a) and [Fig. 3](#) (a). [Fig. 2](#) (b) demonstrates that as rotors 2 and 3 slow down, rotors 1 and 4 rise in speed. This causes the quadcopter to roll along the x-axis and travel on the y-axis. To shift the quadcopter's position along the x-axis, pitching along the y-

axis is required where the speed of rotors 1 and 2 must rise while rotors 3 and 4 must decrease, (and the otherwise) as shown in Fig. 2 (c). Two diagonal rotors with the same direction of rotation increase (decrease) in speed, which allows the quadcopter to vary its heading in the air (yaw) while rotating along the z-axis, as seen in Fig. 2 (d) and Fig. 3 (d) [37].



**Fig. 2.** (a) Hover (b) Roll (c) Pitch (d) Yaw



**Fig. 3.** (a) Hover (b) Roll (c) Pitch (d) Yaw

From Fig. 2 and Fig. 3, it can be seen that a quadcopter has a translational ( $x, y, z$ ) and rotational ( $\phi, \theta, \gamma$ ) coordinate subsystems. By using the Newton-Euler approach to derive a quadcopter dynamic model, the translational dynamic equation of quadcopter can be deduced as follows [36]:

$$m\ddot{\mathbf{r}} = u_T \mathbf{e}_z - m\mathbf{g}\mathbf{e}_z \quad (1)$$

Where  $g$  is the gravitational acceleration,  $m$  is the mass of the quadcopter, and  $\mathbf{e}_z = (0 \ 0 \ 1)^T$  is the unit vector, and  $u_T$  is the sum of the four rotors' thrusts.

$$u_T = \sum_{i=1}^4 F_i = b \sum_{i=1}^4 \Omega_i^2 \quad (2)$$

The thrust force produced by rotor  $i$ ,  $i = 1, 2, 3, 4$  is  $F_i = b \cdot \Omega_i^2$ , where  $b$  is the thrust factor and  $\Omega_i$  is the speed of rotor  $i$ . The body frame's rotation matrices  $R$  is regarded to the earth frames and is given by [38]:

$$R_b^e = \begin{bmatrix} C\theta C\psi & S\phi S\theta C\psi - S\psi C\phi & S\theta C\phi C\psi + S\phi S\psi \\ S\psi C\theta & S\phi S\theta S\psi + C\phi C\psi & S\theta S\psi C\phi - S\phi S\psi \\ -S\theta & S\phi C\theta & C\phi C\theta \end{bmatrix} \quad (3)$$

Where  $C$  (angle) stands for cosine and  $S$  (angle) for sine. The quadcopter is oriented by three Euler angles: roll angle ( $\phi$ ), pitch angle ( $\theta$ ), and yaw angle ( $\psi$ ). This quadcopter's rotational dynamic equation is also provided by [36]:

$$I\dot{\omega} = -\omega \times I\omega - J_r(\omega \times \mathbf{e}_z)(-1)^{i+1}\Omega_r + \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} \quad (4)$$

In this context,  $I$  represents the inertia matrix,  $\omega \times I\omega$  and  $J_r(\omega \times \mathbf{e}_z)(-1)^{i+1}\Omega_r$  denote the gyroscopic effects resulting from rigid body rotation and changes in propeller orientation,

respectively. The control torque,  $\tau$ , is obtained by adjusting the rotor speeds. Based on (1) and (4), below is a representation of the generated translational and rotational equation:

$$\begin{aligned}\ddot{x} &= \frac{1}{m} (S\theta C\phi C\psi + S\phi S\psi) U_1 \\ \ddot{y} &= \frac{1}{m} (S\theta C\psi C\phi + S\phi S\psi) U_1 \\ \ddot{z} &= \frac{1}{m} (C\phi C\theta) U_1 - g \\ \ddot{\phi} &= \frac{1}{I_{xx}} [(I_{yy} - I_{zz}) \dot{\theta} \dot{\psi} - J_r \dot{\theta} \Omega_r + I U_2] \\ \ddot{\theta} &= \frac{1}{I_{yy}} [(I_{zz} - I_{xx}) \dot{\phi} \dot{\psi} + J_r \dot{\phi} \Omega_r + I U_3] \\ \ddot{\psi} &= \frac{1}{I_{zz}} [(I_{xx} - I_{yy}) \dot{\theta} \dot{\phi} + U_4]\end{aligned}\quad (5)$$

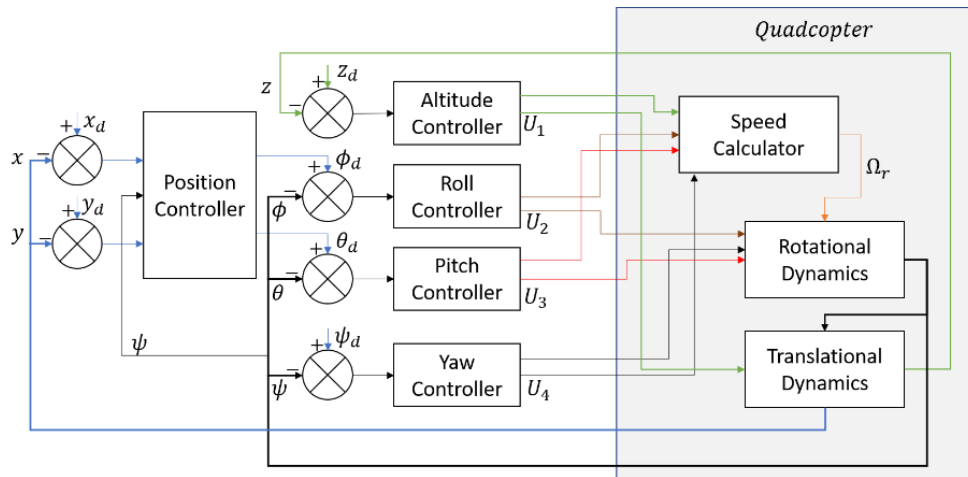
By referring to [39], the parameters are set accordingly for the quadcopter model in this work as tabulated in Table 1.

**Table 1.** Parameters associated with quadcopter model

| Parameters  | Values                               |
|-------------|--------------------------------------|
| $g$         | $9.81 \text{ m.s}^{-2}$              |
| $m$         | $0.5 \text{ kg}$                     |
| $\ell$      | $0.2 \text{ m}$                      |
| $J_x = J_y$ | $4.85 \times 10^{-3} \text{ kg.m}^2$ |
| $J_z$       | $8.81 \times 10^{-3} \text{ kg.m}^2$ |
| $J_r$       | $3.36 \times 10^{-5} \text{ kg.m}^2$ |
| $K_T$       | $2.92 \times 10^{-6} \text{ kg.m}$   |
| $K_d$       | $1.12 \times 10^{-7} \text{ kg.m}^2$ |

## 2.2. Controller's Structure

In this section, the controller structure in a quadcopter system is visualized in block models. The design and methodology of the auto-tuned PID controller using neural network are also presented in this section. Fig. 4 represents the block diagram of overall control structure of quadcopter system which includes the quadcopter model and individual controllers for altitude, attitudes, and position. From Fig. 4, it can be seen that roll and pitch controllers' inputs depending on the position controller's output in a complete quadcopter system.



**Fig. 4.** Block diagram of quadcopter system

### 2.2.1. Altitude Control

To control an altitude of a quadcopter, the error (difference between desired input and the actual output) is taken into a controller for error attenuation, so that the controller will output an adjusted control input,  $U_1$ , for the quadcopter to fly to a desired altitude. Fig. 5 visualizes the general control structure for altitude of quadcopter system.  $\Omega_r$  is a relative speed where in some literatures, it is used as an internal disturbance as included in (5).

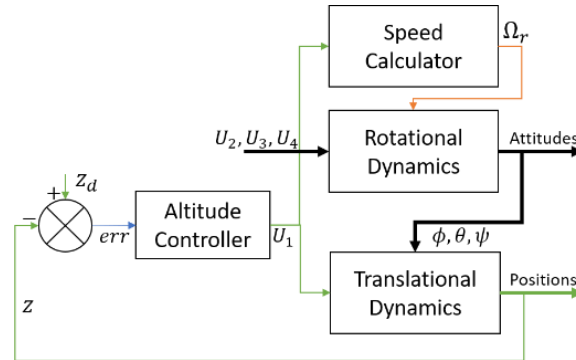


Fig. 5. Block diagram for altitude control

### 2.2.2. Attitude Control

Attitudes in a quadcopter are the Euler angles associated with a quadcopter movement which are so-called, roll ( $\phi$ ), pitch ( $\theta$ ) and yaw ( $\psi$ ) as in Fig. 2 and Fig. 3. To control the three angles, each error is taken to produce the respective control inputs,  $U_2$ ,  $U_3$  and  $U_4$ , for the quadcopter to meet the desired requirements given. Fig. 6 depicted the structure of attitude control in a quadcopter system.

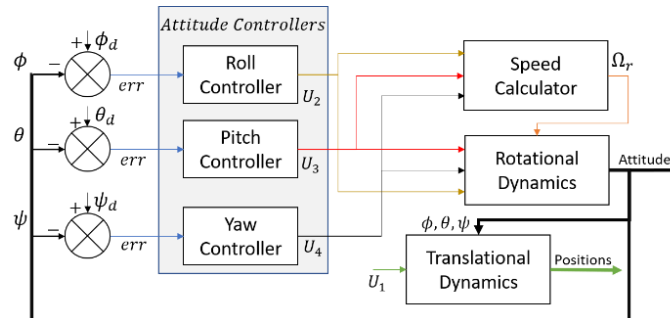
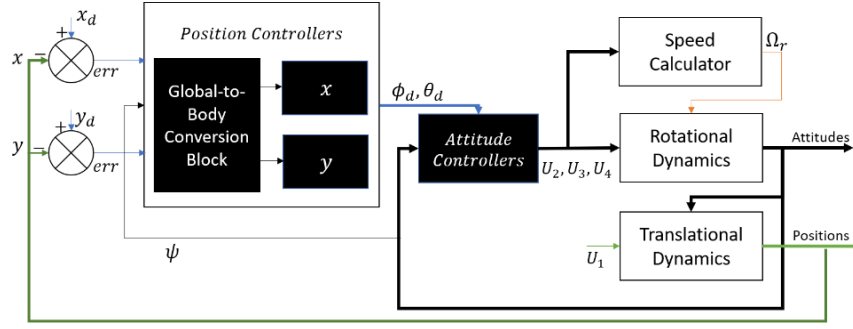


Fig. 6. Block diagram for attitudes control

### 2.2.3. X-Y Position Control

Due to the underactuated and coupling dynamic characteristics of a quadcopter, the position  $x$  and  $y$  cannot be controlled directly. In some research,  $x$  and  $y$  positioning are regarded as the underactuated part [40] of a quadcopter while the attitudes and altitude are called the fully actuated part of a quadcopter. Only through the change of roll and pitch angle movement, the quadcopter can achieve a translational movement in  $x$  and  $y$  axis, and this is the occurrence of coupling dynamic in a quadcopter.

In Fig. 7, the position controller embodied controllers for both  $x$  and  $y$  translational movement of a quadcopter with a conversion block from world frame to body frame. Looking at Fig. 7, the position controller is considered the outer loop in the cascaded loop of controllers in a quadcopter. The position controller also receives input from the observed yaw angle. The reason is that while roll and pitch are relative to the drone's body, the  $x$  and  $y$  position error is relative to the ground or the world reference frame. As a result, pitch doesn't always move the drone in the  $x$ -world direction, and roll doesn't always move the drone in the  $y$ -world direction. Equation (6) is used to build the conversion block for each  $x$  and  $y$  position.



**Fig. 7.** Block diagram for position control

$$\begin{aligned} x^G \cos \psi + y^G \sin \psi &= x^B \\ x^G \sin \psi - y^G \cos \psi &= y^B \end{aligned} \quad (6)$$

Based on (6), if the quadcopter needs to be moved to a very specific location in the room, it will depend on how it is rotated or its yaw angle and it needs to know whether roll or pitch or a mix of the two will be necessary to do that. The outputs from (6) are then fed into controllers to produce the desired roll and pitch angle as in (7), for the inner loop controllers (the attitudes and altitude controllers).

$$\begin{aligned} \phi_d &= K_p(y_d^B - y^B) + K_i \int_0^t (y_d^B - y^B) dt + K_d \frac{d(y_d^B - y^B)}{dt} \\ \theta_d &= K_p(x_d^B - x^B) + K_i \int_0^t (x_d^B - x^B) dt + K_d \frac{d(x_d^B - x^B)}{dt} \end{aligned} \quad (7)$$

### 2.3. PID Controller

The PID control law involved in this work is presented mathematically in (8). Direct inputs are given to the position, altitude and yaw controllers while roll and pitch controllers receive inputs from the position controller. A PID controller works by reducing the errors resulting from the difference between the input reference and the actual output of the quadcopter states. The proportional term creates an instant multiplied output to align the process value with the input reference. The integral term accumulates error over simulation time where more effect is needed until the process value reaches the setpoint. The derivative term helps to predict the future of the process value by multiplying with the ramp rate of the process value to prevent overshooting if the ramp rate is too fast [41].

$$\begin{aligned} U_1 &= K_P e_z + K_I \int_0^t e_z dt + K_D \frac{de_z}{dt} \\ U_2 &= K_P e_\phi + K_I \int_0^t e_\phi dt + K_D \frac{de_\phi}{dt} \\ U_3 &= K_P e_\theta + K_I \int_0^t e_\theta dt + K_D \frac{de_\theta}{dt} \\ U_4 &= K_P e_\psi + K_I \int_0^t e_\psi dt + K_D \frac{de_\psi}{dt} \end{aligned} \quad (8)$$

From the generation of offline dataset to the network training, simulations are done using MATLAB/Simulink software including the building of quadcopter model. Table 2 shows the manual tuned PID gains that act as PID initialization for the dataset generation which later to be used for neural network training. The gains were obtained by following the PID manual tuning method provided by [42]. In general, the performance of a manually tuned PID controller is excellent,

particularly for simple flight inputs such as step inputs. However, its performance may degrade when faced with more complex trajectory tracking patterns. By leveraging the analysis capabilities of a neural network on various combinations of flight outputs and gains, it is possible to obtain a more consistent set of PID gains.

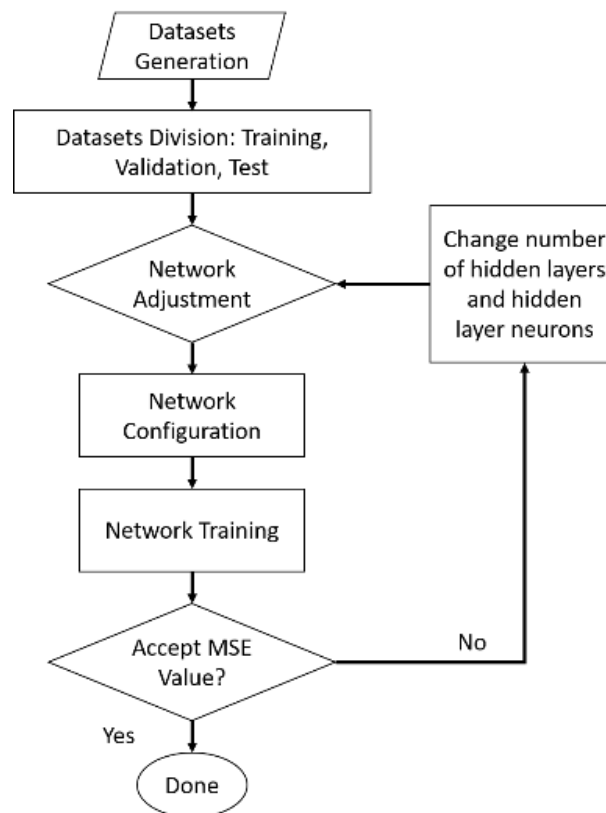
**Table 2.** The best gains obtained from PID manual tune to generate a new dataset

| Parameters | $K_p$   | $K_i$   | $K_d$  |
|------------|---------|---------|--------|
| Roll       | 50      | 4.1531  | 3.1645 |
| Pitch      | 17.4897 | 3.9749  | 3.9290 |
| Yaw        | 33.7736 | 2.4542  | 0.9508 |
| Thrust     | 50      | 30.3777 | 8.8697 |

The dataset is generated by simulating the quadcopter system by using the range of PID gains set differently for each of the gains, by referring to the optimized gains as PID controllers' gains initialization. There are seven items that are included as inputs to the ANN, which are the reference, control input, PID controller error [32], rise time ( $T_r$ ), settling time ( $T_s$ ), overshoot time ( $T_o$ ) and overshoot period ( $O_p$ ) [34], for each of the parameters in Table 2. 500 lines of PID gains with their corresponding seven items are registered in each dataset ( $\phi, \theta, \gamma, z$ ) for ANN training in the next section.

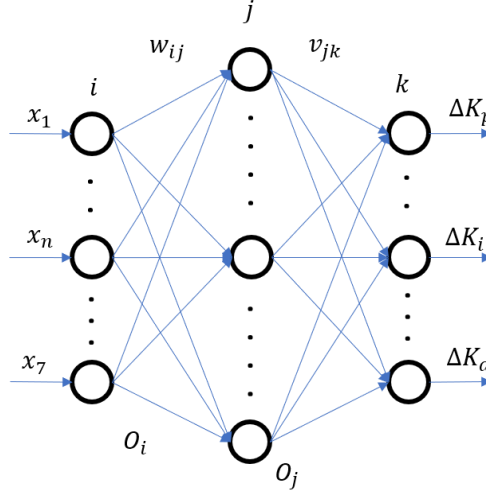
#### 2.4. Neural Network Auto-Tuner PID Controller

A feedforward neural network is developed and explained in detail in this sub-section. An artificial neural network (ANN) is an imitation of functions of human brain cells which involves axons and dendrites that perform information transferring and interpreting, using inputs obtained from the human's five senses. In this work, the newly developed neural network needs to be trained first, as how a human child being trained to perform certain actions. The datasets generated will be used to train the network. Each dataset prepared has seven inputs (ref, u, e,  $T_r$ ,  $T_s$ ,  $T_o$ ,  $O_p$ ) and 3 targeted outputs ( $K_p$ ,  $K_i$ ,  $K_d$ ) for the training.



**Fig. 8.** Flowchart of network creation

Fig. 8 briefly visualizes the process of network creation starting from the use of datasets for network training. Fig. 9 shows the network structure, where  $x$  denotes the 7 inputs into the network and  $\Delta K_{p,i,d}$  denotes the 3 calculated outputs of the network  $i, j$  and  $k$  denote the neurons of input, hidden and output layer respectively. The weight matrices of input layer to hidden layer and from hidden layer to output layer are represented by  $w_{ij}$  and  $v_{jk}$  respectively. Output from each layer is denoted by  $O_i$ ,  $O_j$  and  $O_k$ . Equations (9) to (14) below are the mathematical model of feedforward ANN [43].



**Fig. 9.** The general structure of feedforward ANN

Input to the hidden layer.

$$O_i = \sum_{i=1}^{16} w_{ij} x_i \quad (9)$$

Output of hidden layer.

$$O_j = \frac{1}{1 + e^{-O_i}} \quad (10)$$

Input to the output layer.

$$r = \sum_{j=1}^n v_{jk} O_j \quad (11)$$

Output of output layer.

$$\Delta K_G = \frac{1}{1 + e^{-r}} \quad (12)$$

Error from training.

$$e = \Delta K_G - T \quad (13)$$

Mean Square Error to evaluate the training performance.

$$MSE = \frac{\sum_{i=1}^n e^2}{i} \quad (14)$$

$\therefore T$  = Target output from dataset.

The ANN used in this work is also considered as a supervised learning process because targeted outputs are used as goals for the ANN to accomplish through training. The actual output from ANN training is compared with the targeted output to get the errors as in (13) and the training performances is evaluated using MSE of the cumulative errors and the regression values. Fig. 10 shows the block diagram of the whole system.

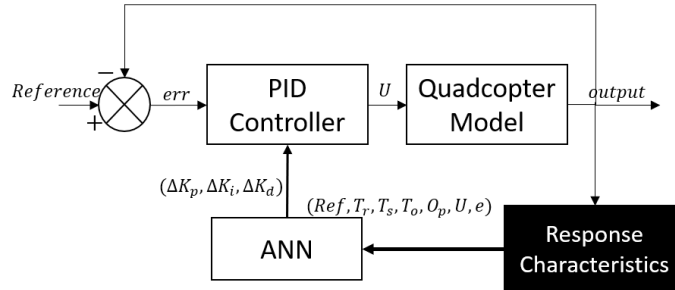
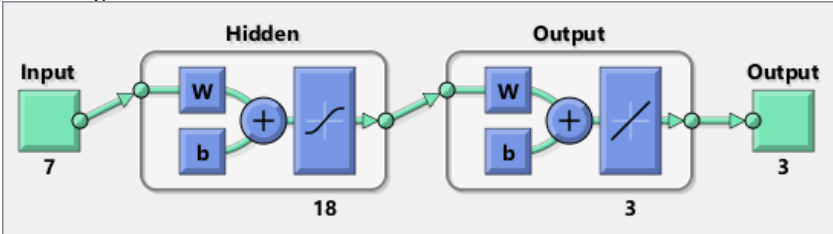


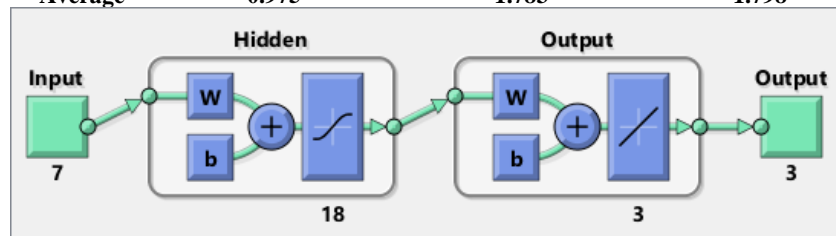
Fig. 10. Block diagram of PIDNN-quadcopter system

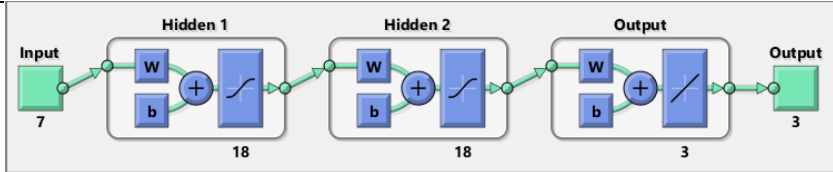
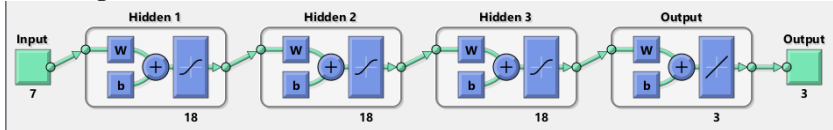
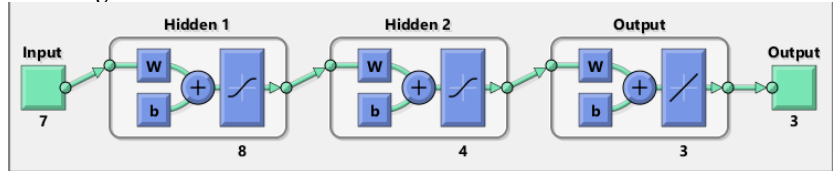
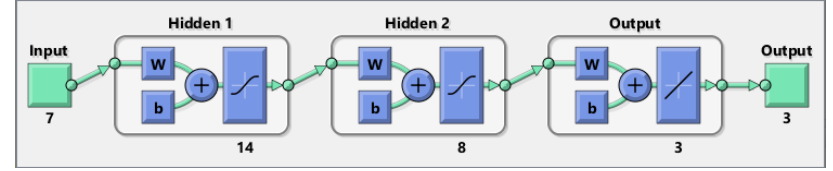
Different network cases are being tabulated in Table 3 to identify the best network to be used in this work. Case 1 to Case 3 represent the regression values obtained from networks with different numbers of hidden layers but with same neurons. Case 4 to Case 6 represent the regression values obtained from networks with the same number of hidden layers but different neurons. Each dataset was randomly divided as follows: 70% of the data for the training set, 15% for a network validation set, and 15% for evaluating the network performance (test set). Splitting the dataset into these three subsets helps ensure that the model learns effectively, generalizes well to new data, and provides reliable performance estimates. The test set is used to assess the final performance of the trained model on unseen data and the best network will depend on MSE values from the test set.

Based on Table 3, it is observed that Case 6 produces the most acceptable result with the lowest MSE value of the test set. The trained network from Case 6 is chosen to produce results in the next section. As conclusion from this section, the ANN contains three hidden layers and uses 300 epochs. The Levenberg-Marquardt training method was selected because of its widespread use in the field of artificial neural network (ANN) research. The minimum gradient is set to be  $1e-7$  and the training performance is evaluated using MSE with a target of  $1e-6$ . All these specifications were finalized after being adjusted several times according to the network performance.

Table 3. ANN training cases

| Cases  | Hidden Layers | MSE Performance Values   |                  |            |        |
|--------|---------------|--|------------------|------------|--------|
|        |               | Train (70%)  | Validation (15%) | Test (15%) |        |
| Case 1 | [18]          | Z  | 1.277            | 2.161      | 2.378  |
|        |               | Roll   | 1.288            | 2.495      | 2.008  |
|        |               | Pitch  | 0.4008           | 0.7267     | 0.8348 |
|        |               | Yaw  | 0.934            | 1.751      | 1.973  |
|        |               | Average  | 0.975            | 1.783      | 1.798  |
|        |               |  |                  |            |        |
| Case 2 | [18 18]       | Z  | 0.2461           | 2.956      | 3.194  |
|        |               | Roll   | 0.06229          | 2.803      | 1.871  |
|        |               | Pitch  | 0.05212          | 0.835      | 1.031  |
|        |               | Yaw  | 0.3616           | 2.104      | 1.754  |
|        |               | Average  | 0.181            | 2.175      | 1.963  |



| Cases  | Hidden Layers | MSE Performance Values   |                  |            |        |
|--------|---------------|--|------------------|------------|--------|
|        |               | Train (70%)  | Validation (15%) | Test (15%) |        |
| Case 3 | [18 18 18]    |    |                  |            |        |
|        |               | Z  | 0.02202          | 2.683      | 3.132  |
|        |               | Roll   | 0.01701          | 1.919      | 2.745  |
|        |               | Pitch  | 0.00595          | 0.9223     | 1.105  |
|        |               | Yaw  | 0.1771           | 1.8        | 2.155  |
|        |               | Average  | 0.056            | 1.831      | 2.284  |
| Case 4 | [8 4]         |    |                  |            |        |
|        |               | Z  | 1.512            | 2.127      | 2.52   |
|        |               | Roll   | 1.481            | 2.424      | 1.706  |
|        |               | Pitch  | 0.5703           | 0.8975     | 0.8517 |
|        |               | Yaw  | 1.075            | 1.773      | 2.269  |
|        |               | Average  | 1.160            | 1.805      | 1.837  |
| Case 5 | [14 8]        |   |                  |            |        |
|        |               | Z  | 0.8              | 2.484      | 3.117  |
|        |               | Roll   | 0.622            | 2.321      | 1.888  |
|        |               | Pitch  | 0.2112           | 0.8073     | 0.7759 |
|        |               | Yaw  | 0.6495           | 2.08       | 1.827  |
|        |               | Average  | 0.571            | 1.923      | 1.902  |
| Case 6 | [18 9]        |  |                  |            |        |
|        |               | Z  | 0.6719           | 1.991      | 2.545  |
|        |               | Roll   | 0.4029           | 2.181      | 2.281  |
|        |               | Pitch  | 0.1676           | 0.8294     | 0.8815 |
|        |               | Yaw  | 0.6683           | 1.937      | 1.537  |
|        |               | Average  | 0.478            | 1.7346     | 1.811  |

### 3. Results and Discussion

The MATLAB/Simulink environment was used to execute the simulations. The simulation is conducted on a personal computer with 8GB RAM which is ideal to run MATLAB/Simulink software for quadcopter system verification, together with the designed controller. MATLAB runs for the dataset and the ANN while Simulink simulates the whole quadcopter system including designed controllers. The quadcopter model is built and simulated using the equations provided in section 2.1.

In this section, results are presented through altitude and attitude stabilization using step input and multi-level tracking, and position tracking. Perturbations are also added during multi-level altitude tracking to assess the reliability of the designed controller compared to manual tuned PID. The chosen

trained network from the previous section is used to produce final and most reliable results in this section.

### 3.1. Altitude and Attitude Stabilization

Looking at Fig. 11, Fig. 12, Fig. 13, until Fig. 14, the auto-tuned PID controller has greatly improved the quadcopter performance compared to the manual tuned PID controller in terms of reducing overshoot, rise time and settling time. The proposed controller has demonstrated its efficiency in stabilizing the altitude and attitude of a quadcopter, effectively controlling its states under the influence of step inputs.

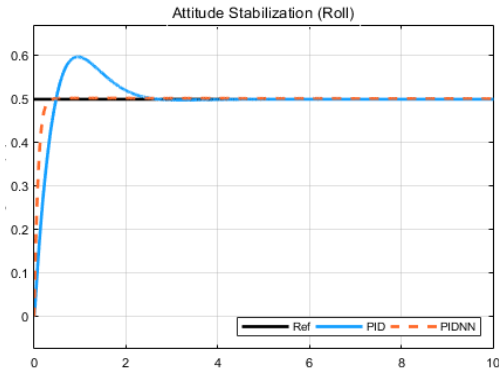


Fig. 11. PIDNN for roll angle tracking

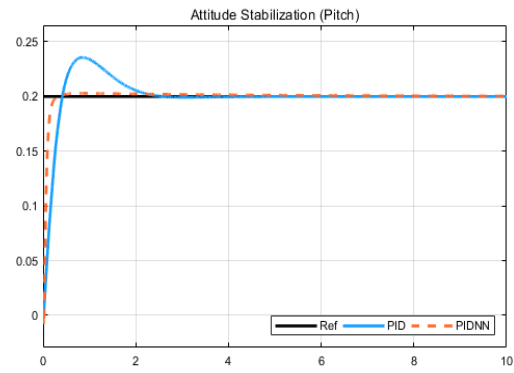


Fig. 12. PIDNN for pitch angle tracking

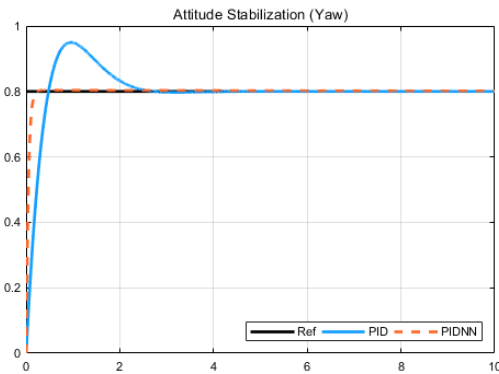


Fig. 13. PIDNN for yaw angle tracking

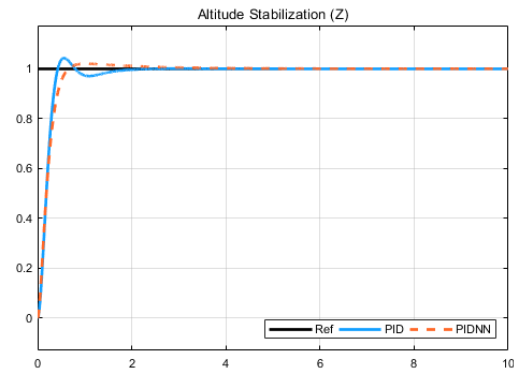


Fig. 14. PIDNN for altitude tracking

### 3.2. Altitude and Attitude Multi-Level Tracking

A linear quadcopter model is also tested using the auto-tuned PID gains where all attitude angles are set to zero. In Fig. 15, the result shows lower overshoot with longer settling time compared to the manual tuned PID gains. Both controllers successfully track the desired input reference, although differing performance characteristics are observed.

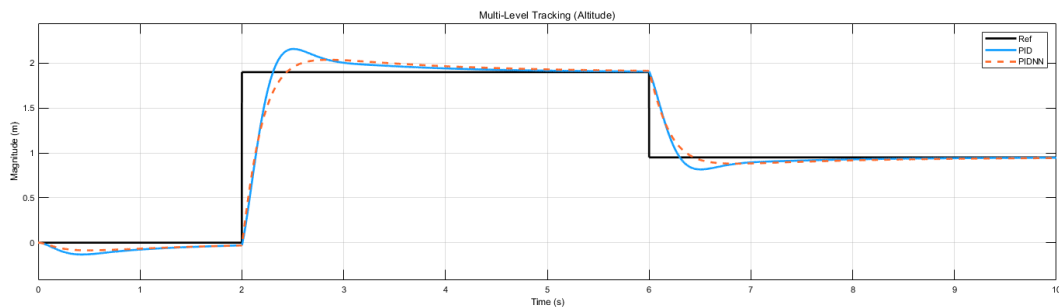


Fig. 15. Multi-level simulation of linear-model quadcopter (altitude tracking)

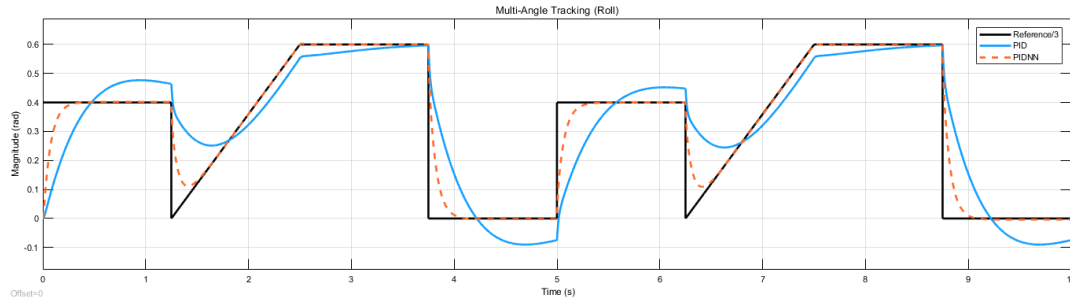


Fig. 16. Multi-angle tracking of roll

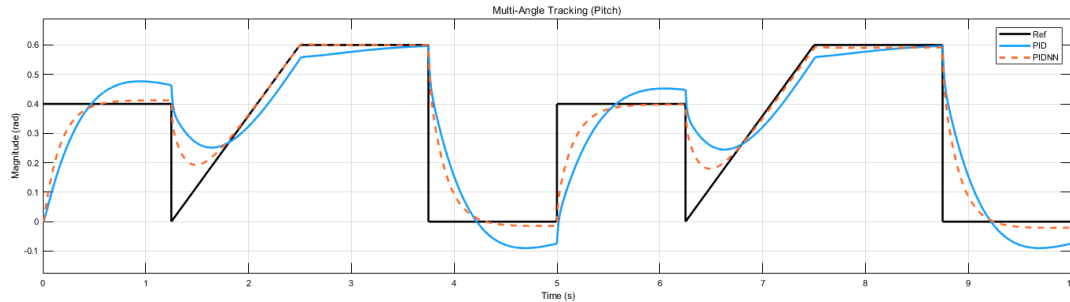


Fig. 17. Multi-angle tracking of pitch

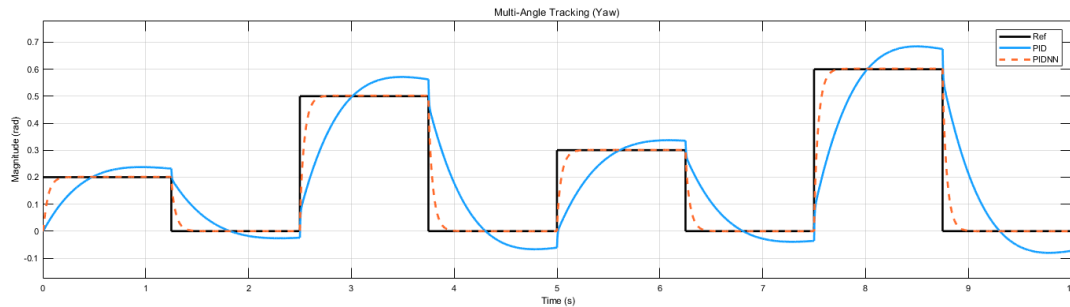


Fig. 18. Multi-angle tracking of yaw

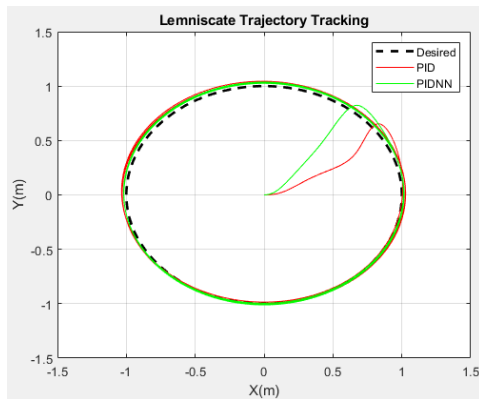
Looking at Fig. 16, Fig. 17, Fig. 18, all multi-angle tracking of roll, pitch and yaw using the resulting gains from the designed controller can follow the input reference much better than the normal PID controller. However, it is difficult for both controllers to follow a sudden change in reference such as 1-2 seconds and 6-7 seconds in Fig. 16 and Fig. 17 while a constant reference is easier to track.

### 3.3. Position Tracking

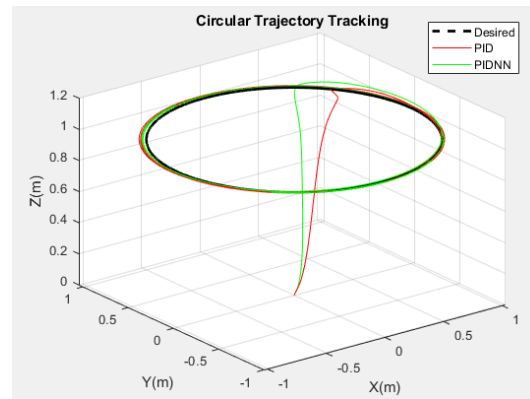
Fig. 19, Fig. 20, Fig. 21 until Fig. 22 depicted the position tracking of quadcopter in circular and lemniscate trajectory tracking for 20 seconds of simulation for a complete pattern. As can be seen, steady-state error occurs in both simulations, where it requires a longer time to settle following the desired path. The new gains provide a better overshoot and smaller steady-state error in position tracking as compared to old ones. Table 4, Table 5, Table 6, until Table 7 presents the controllers error based on index performances of IAE, ISE, ITAE and ITSE for X-position, Y-position, roll-angle, and pitch-angle, respectively. Overall error shows that the new gains (PIDNN) are associated with much lower error values if compared to the manual tuned gains.

Table 4. The error performance values comparison of trajectory tracking (X-position)

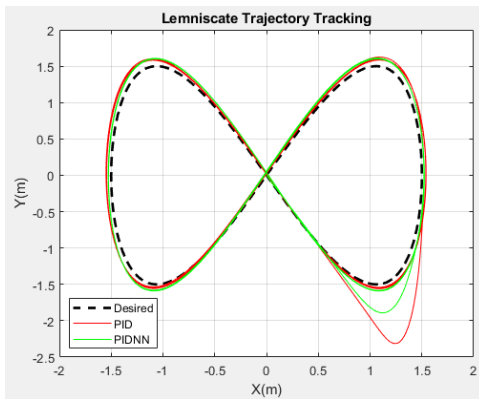
| X          | PID    |         |       |        | PIDNN  |          |       |        |
|------------|--------|---------|-------|--------|--------|----------|-------|--------|
|            | IAE    | ISE     | ITAE  | ITSE   | IAE    | ISE      | ITAE  | ITSE   |
| Circular   | 0.6074 | 0.01722 | 18.22 | 0.5165 | 0.3468 | 0.005499 | 10.4  | 0.165  |
| Lemniscate | 0.8981 | 0.03543 | 26.94 | 1.063  | 0.5122 | 0.01156  | 15.37 | 0.3468 |



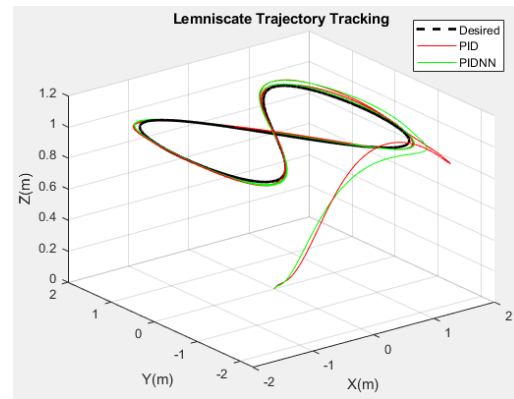
**Fig. 19.** PIDNN for circular position tracking (2D)



**Fig. 20.** PIDNN for circular position tracking (3D)



**Fig. 21.** PIDNN for lemniscate position tracking (2D)



**Fig. 22.** PIDNN for lemniscate position tracking (3D)

**Table 5.** The error performance values comparison of trajectory tracking (Y-position)

| Y          | PID   |        |       |       | PIDNN  |        |       |       |
|------------|-------|--------|-------|-------|--------|--------|-------|-------|
|            | IAE   | ISE    | ITAE  | ITSE  | IAE    | ISE    | ITAE  | ITSE  |
| Circular   | 1.078 | 0.423  | 32.34 | 12.69 | 0.7949 | 0.2687 | 23.85 | 8.061 |
| Lemniscate | 2.162 | 0.5207 | 64.85 | 15.62 | 2.155  | 0.2324 | 64.66 | 6.971 |

**Table 6.** The error performance values comparison of trajectory tracking (roll angle)

| $\phi$     | PID    |        |       |       | PIDNN  |         |       |        |
|------------|--------|--------|-------|-------|--------|---------|-------|--------|
|            | IAE    | ISE    | ITAE  | ITSE  | IAE    | ISE     | ITAE  | ITSE   |
| Circular   | 0.6398 | 0.1064 | 19.19 | 3.191 | 0.2288 | 0.02646 | 6.863 | 0.7939 |
| Lemniscate | 3.189  | 0.4324 | 95.67 | 12.97 | 0.9064 | 0.04183 | 27.19 | 1.255  |

**Table 7.** The error performance values comparison of trajectory tracking (pitch angle)

| $\theta$   | PID    |         |       |        | PIDNN  |         |       |        |
|------------|--------|---------|-------|--------|--------|---------|-------|--------|
|            | IAE    | ISE     | ITAE  | ITSE   | IAE    | ISE     | ITAE  | ITSE   |
| Circular   | 0.4637 | 0.02515 | 13.91 | 0.7546 | 0.3824 | 0.01674 | 11.47 | 0.5023 |
| Lemniscate | 0.6451 | 0.03438 | 19.35 | 1.032  | 0.5469 | 0.02438 | 16.41 | 0.7313 |

### 3.4. Robustness Evaluation during Presence of Perturbation

To further evaluate the effectiveness of the designed controller, a wind gust disturbance is added into the altitude of the quadcopter model as shown in Fig. 23 (a). The performance of both controllers

in facing perturbation is almost the same based on Fig. 24. The auto-tuned gains (PIDNN) can handle rising and falling edge better than the manual tuned gains and have better convergence towards the setpoint. To further analyze, Root Mean Square Error (RMSE) evaluation is introduced in Table 8. Based on the numerical evaluation, the auto-tuned PID controller performs 1.7% better than the manual tuned PID controller. The steps taken from generating the datasets to network training have produced a more adaptive PID controller gains that can homogenously respond to different response characteristics of the quadcopter states in an offline manner.

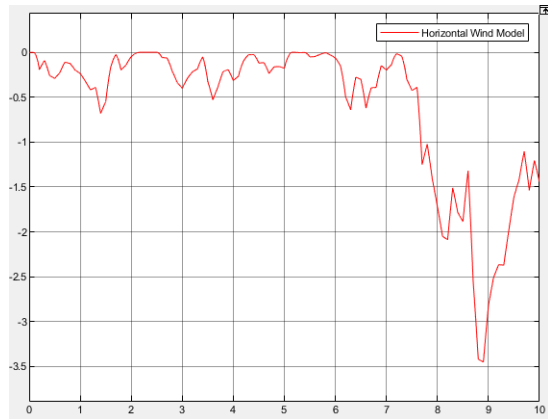


Fig. 23. Disturbance model

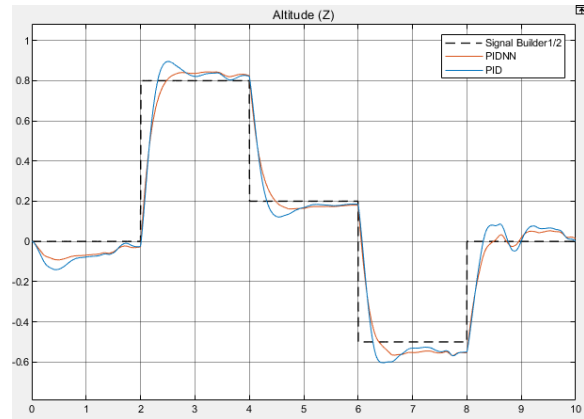


Fig. 24. Multistep altitude during perturbation

Table 8. The RMSE evaluation during perturbation

| PID    | PIDNN  |
|--------|--------|
| 0.4472 | 0.4396 |

All the results have shown minor improvements being made by the designed controller if compared to the manual tuned one. This work has successfully presented the design and performance of the auto-tuned PID controller using neural network in comparison to manual tuned PID controller. The concept of generating auto-tuned PID gains using an artificial neural network (ANN), trained through analysis of various response characteristics, has been successfully demonstrated in this work. To enhance the effectiveness of the network, additional datasets with diverse setpoints should be generated. Furthermore, integrating real-time feedback from the quadcopter system could further enhance the performance of the designed controller.

#### 4. Conclusion

In conclusion, this study has demonstrated the performance of auto-tuned neural network PID controller in controlling a quadcopter UAV, with a comparative analysis conducted against a manually tuned PID controller. Unlike traditional PID manual tuning, which lacks comprehensive analysis, the use of a neural network enables the analysis of an entire dataset containing 500 simulation results with varying PID gains, leading to better auto-tuning of the controller. The dataset was formed using an initialization of PID gains obtained from manual tuning, and multiple networks with different architectures were evaluated using MSE performance. The results revealed that the auto-tuned PID gains provided reduced overshoot and very low steady-state error compared to manually tuned PID gains, albeit with a sacrifice in settling time. During perturbation, the auto-tuned PID controller outperformed the manual tuned PID controller by 1.7%. Notably, improvements were observed primarily during lower setpoints, suggesting the need for generating datasets according to various setpoints to enhance the adaptability of the neural network. For future research, it is recommended to widen the analysis spectrum of the neural network by generating datasets with diverse setpoints.

Additionally, applying the ANN-driven PID controller to a real-time feedback quadcopter system could enhance the controller's functionality in stabilizing the quadcopter during flight. Overall, this study contributes valuable insights into the effectiveness of auto-tuned neural network PID controllers and highlights avenues for further exploration in adaptive control strategies for quadcopter systems.

**Author Contribution:** All authors contributed equally to the main contributor to this paper. All authors read and approved the final paper.

**Acknowledgment:** The authors would like to thank the Ministry of Higher Education (MOHE) through Fundamental Research Grant Scheme (FRGS/1/2021/TK0/UTM/02/56) and Universiti Teknologi Malaysia through UTMFR (Q.J130000.3823.22H67) for supporting this research.

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

## References

- [1] M. Moshref-javadi and M. Winkenbach, "Applications and Research avenues for drone-based models in logistics : A classification and review," *Expert Systems with Applications*, vol. 177, p. 114854, 2021, <https://doi.org/10.1016/j.eswa.2021.114854>.
- [2] B. Chamberlain and W. Sheikh, "Design and Implementation of a Quadcopter Drone Control System for Photography Applications," *2022 Intermountain Engineering, Technology and Computing (IETC)*, pp. 1-7, 2022, <https://doi.org/10.1109/IETC54973.2022.9796735>.
- [3] Y. Ko, J. Kim, D. G. Duguma, P. V. Astillo, and I. You, G. Pau, "Drone Secure Communication Protocol for Future Sensitive Applications in Military Zone," *Sensors*, vol. 21, no. 6, p. 2057, 2021, <https://doi.org/10.3390/s21062057>.
- [4] A. Jaishwal and V. M. Lakshe, "Weather Station Quadcopter Using Arduino with NRF24L01 and GPS Module," *International Research Journal of Engineering and Technology*, vol. 6, no. 3, pp. 4690–4691, 2019, <https://www.irjet.net/archives/V6/i3/IRJET-V6I31202.pdf>.
- [5] M. A. M. Basri, A. R. Husain, and K. A. Danapalasingam, "Enhanced Backstepping Controller Design with Application to Autonomous Quadrotor Unmanned Aerial Vehicle," *Journal of Intelligent & Robotic Systems*, vol. 79, pp. 295–321, 2015, <https://doi.org/10.1007/s10846-014-0072-3>.
- [6] Z. He and L. Zhao, "A simple attitude control of quadrotor helicopter based on Ziegler-Nichols rules for tuning pd parameters," *The Scientific World Journal*, vol. 2014, 2014, <https://doi.org/10.1155/2014/280180>.
- [7] S. Khatoon, M. Shahid, Ibraheem and H. Chaudhary, "Dynamic modeling and stabilization of quadrotor using PID controller," *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 746-750, 2014, <https://doi.org/10.1109/ICACCI.2014.6968383>.
- [8] R. Roy, M. Islam, N. Sadman, M. A. P. Mahmud, K. D. Gupta, and M. M. Ahsan, "A Review on Comparative Remarks, Performance Evaluation and Improvement Strategies of Quadrotor Controllers," *Technologies*, vol. 9, no. 2, p. 37, 2021, <https://doi.org/10.3390/technologies9020037>.
- [9] A. Latif, K. Shankar, P. T. Nguyen, "Legged Fire Fighter Robot Movement Using PID," *Journal of Robotics and Control*, vol. 1, no. 1, pp. 15–18, 2020, <https://doi.org/10.18196/jrc.1104>.
- [10] M. Mahmud, S. M. A. Motakabber, A. H. M. Z. Alam, and A. N. Nordin, "Control BLDC motor speed using PID controller," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 3, pp. 477-481, 2020, <https://dx.doi.org/10.14569/IJACSA.2020.0110359>.
- [11] J. D. S. G. Barros, L. A. Rossi, Z. M. D. Souza, "PID temperature controller in pig nursery: spatial characterization of thermal environment," *International Journal of Biometeorology*, vol. 62, pp. 773-781, 2018, <https://doi.org/10.1007/s00484-017-1479-x>.
- [12] M. R. Habib *et al.*, "PID Controller Based Automatic Solar PowerDriven Grass Cutting Machine," *2019 International Conference on Computer, Communication, Chemical, Materials and Electronic Engineering (IC4ME2)*, pp. 1-4, 2019, <https://doi.org/10.1109/IC4ME247184.2019.9036513>.
- [13] N. Bao, X. Ran, Z. Wu, Y. Xue and K. Wang, "Research on attitude controller of quadcopter based on

- cascade PID control algorithm," *2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, pp. 1493-1497, 2017, <https://doi.org/10.1109/ITNEC.2017.8285044>.
- [14] C. S. Subudhi and D. Ezhilarasi, "Modeling and Trajectory Tracking with Cascaded PD Controller for Quadrotor," *Procedia Computer Science*, vol. 133, pp. 952-959, 2018, <https://doi.org/10.1016/j.procs.2018.07.082>.
- [15] S. Abdelhay and A. Zakriti, "Modeling of a Quadcopter Trajectory Tracking System Using PID Controller," *Procedia Manufacturing*, pp. 564-571, 2019, <https://doi.org/10.1016/j.promfg.2019.02.253>.
- [16] P. Burggräf, A. R. P. Martínez, H. Roth, and J. Wagner, "Quadrotors in factory applications: design and implementation of the quadrotor's P-PID cascade control system: Modeling and implementation," *SN Applied Sciences*, vol. 1, no. 7, p. 722, 2019, <https://doi.org/10.1007/s42452-019-0698-7>.
- [17] J. Moreno-Valenzuela, R. Pérez-Alcocer, M. Guerrero-Medina and A. Dzul, "Nonlinear PID-Type Controller for Quadrotor Trajectory Tracking," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 5, pp. 2436-2447, 2018, <https://doi.org/10.1109/TMECH.2018.2855161>.
- [18] A. A. Najm and I. K. Ibraheem, "Nonlinear PID controller design for a 6-DOF UAV quadrotor system," *Engineering Science and Technology, an International Journal*, vol. 22, no. 4, pp. 1087-1097, 2019, <https://doi.org/10.1016/j.jestch.2019.02.005>.
- [19] A. Noordin, M. A. M. Basri, and Z. Mohamed, "Simulation and experimental study on pid control of a quadrotor MAV with perturbation," *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 5, pp. 1811-1818, 2020, <https://doi.org/10.11591/eei.v9i5.2158>.
- [20] Q. Jiao, J. Liu, Y. Zhang and W. Lian, "Analysis and design the controller for quadrotors based on PID control method," *2018 33rd Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, pp. 88-92, 2018, <https://doi.org/10.1109/YAC.2018.8406352>.
- [21] A. Sheta, M. Braik, D. R. Maddi, A. Mahdy, S. Aljahdali, and H. Turabieh, "Optimization of PID Controller to Stabilize Quadcopter Movements using Meta-Heuristic Search Algorithms," *Applied Science*, vol. 11, no. 14, p. 6492, 2021, <https://doi.org/10.3390/app11146492>.
- [22] I. P. Canal, M. M. R. Pérez, and M. D. Campos, "Ziegler – Nichols Customization for Quadrotor Attitude Control under Empty and Full Loading Conditions," *Computer Modeling in Engineering & Sciences*, vol. 125, no. 1, pp. 65-75, 2020, <https://doi.org/10.32604/cmes.2020.010741>.
- [23] A. Noordin, M. A. M. Basri, Z. Mohamed, and A. F. Z. Abidin, "Modelling and PSO fine-tuned PID control of quadrotor UAV," *International Journal Advanced Science Engineering Information Technology*, vol. 7, no. 4, pp. 1367-1373, 2017, <https://doi.org/10.18517/ijaseit.7.4.3141>.
- [24] J. A. Cárdenas, U. E. Carrero, E. C. Camacho, and J. M. Calderón, "Optimal PID  $\phi$  axis Control for UAV Quadrotor based on Multi-Objective PSO," *IFAC-PapersOnLine*, vol. 55, no. 14, pp. 101-106, 2022, <https://doi.org/10.1016/j.ifacol.2022.07.590>.
- [25] N. H. Sahrir and M. A. M. Basri, "PSO-PID Controller for Quadcopter UAV: Index Performance Comparison," *Arabian Journal for Science and Engineering*, vol. 48, pp. 15241-15255, 2023, <https://doi.org/10.1007/s13369-023-08088-x>.
- [26] G. Sonugür, C. O. Gökçe, Y. B. Koca, Ş. S. Inci, and Z. Keleş, "Particle Swarm Optimization Based Optimal Pid Controller for Quadcopters," *Comptes rendus de l'Académie bulgare des Sciences*, vol. 74, no. 12, pp. 1806-1814, 2021, <https://doi.org/10.7546/CRABS.2021.12.11>.
- [27] I. Siti, M. Mjahed, H. Ayad, and A. El Kari, "New Trajectory Tracking Approach for a Quadcopter Using Genetic Algorithm and Reference Model Methods," *Applied Science*, vol. 9, no. 9, p. 1780, 2019, <https://doi.org/10.3390/app9091780>.
- [28] M. F. Q. Say, E. Sybingco, A. A. Bandala, R. R. P. Vicerra and A. Y. Chua, "A Genetic Algorithm Approach to PID Tuning of a Quadcopter UAV Model," *2021 IEEE/SICE International Symposium on System Integration (SII)*, pp. 675-678, 2021, <https://doi.org/10.1109/IEEECONF49454.2021.9382697>.
- [29] S. Imane, M. Mostafa, A. Hassan and E. K. Abdeljalil, "Control of a quadcopter using reference model and genetic algorithm methods," *2015 Third World Conference on Complex Systems (WCCS)*, pp. 1-6, 2015, <https://doi.org/10.1109/ICoCS.2015.7483296>.

- 
- [30] I. E. Hajjami and B. Benhala, "Simulation Experiments of Different Metaheuristics Algorithms using Benchmark Functions: A Performance Study," *2022 International Conference on Intelligent Systems and Computer Vision (ISCV)*, pp. 1-6, 2022, <https://doi.org/10.1109/ISCV54655.2022.9806089>.
- [31] C. B. Jabeur and H. Seddik, "Neural networks on-line optimized PID controller with wind gust rejection for a," *International Review of Applied Sciences and Engineering*, vol. 13, no. 2, pp. 133-147, 2022, <https://doi.org/10.1556/1848.2021.00325>.
- [32] S. Bari, S. S. Zehra Hamdani, H. U. Khan, M. u. Rehman and H. Khan, "Artificial Neural Network Based Self-Tuned PID Controller for Flight Control of Quadcopter," *2019 International Conference on Engineering and Emerging Technologies (ICEET)*, pp. 1-5, 2019, <https://doi.org/10.1109/CEET1.2019.8711864>.
- [33] J. Gómez-Avila, C. López-Franco, A. Y. Alanis and N. Arana-Daniel, "Control of Quadrotor using a Neural Network based PID," *2018 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, pp. 1-6, 2018, <https://doi.org/10.1109/LA-CCI.2018.8625222>.
- [34] O. Rodríguez-Abreo, J. Rodríguez-Reséndiz, C. Fuentes-Silva, R. Hernández-Alvarado and M. D. C. P. T. Falcón, "Self-Tuning Neural Network PID With Dynamic Response Control," *IEEE Access*, vol. 9, pp. 65206-65215, 2021, <https://doi.org/10.1109/ACCESS.2021.3075452>.
- [35] K. M. Thu and A. I. Gavrilov, "Designing and Modeling of Quadcopter Control System Using L1 Adaptive Control," *Procedia Computer Science*, vol. 103, pp. 528-535, 2017, <https://doi.org/10.1016/j.procs.2017.01.046>.
- [36] M. A. Mohd Basri, A. R. Husain, and K. A. Danapalasingam, "Enhanced Backstepping Controller Design with Application to Autonomous Quadrotor Unmanned Aerial Vehicle," *Journal of Intelligent & Robotic Systems*, vol. 79, pp. 295-321, 2015, <https://doi.org/10.1007/s10846-014-0072-3>.
- [37] D. Domingos, G. Camargo, and F. Gomide, "Autonomous Fuzzy Control and Navigation of Quadcopters," *IFAC-PapersOnLine*, vol. 49, no. 5, pp. 73-78, 2016, <https://doi.org/10.1016/j.ifacol.2016.07.092>.
- [38] J. Kim, S. A. Gadsden and S. A. Wilkerson, "A Comprehensive Survey of Control Strategies for Autonomous Quadrotors," *Canadian Journal of Electrical and Computer Engineering*, vol. 43, no. 1, pp. 3-16, 2020, <https://doi.org/10.1109/CJECE.2019.2920938>.
- [39] H. Voos, "Nonlinear control of a quadrotor micro-UAV using feedback-linearization," *2009 IEEE International Conference on Mechatronics*, pp. 1-6, 2009, <https://doi.org/10.1109/ICMECH.2009.4957154>.
- [40] A. Eltayeb, M. F. Rahmat, M. A. M. Basri, M. A. M. Eltoum and S. El-Ferik, "An Improved Design of an Adaptive Sliding Mode Controller for Chattering Attenuation and Trajectory Tracking of the Quadcopter UAV," *IEEE Access*, vol. 8, pp. 205968-205979, 2020, <https://doi.org/10.1109/ACCESS.2020.3037557>.
- [41] M. F. Shehzad, A. Bilal and H. Ahmad, "Position & Attitude Control of an Aerial Robot (Quadrotor) With Intelligent PID and State feedback LQR Controller: A Comparative Approach," *2019 16th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, pp. 340-346, 2019, <https://doi.org/10.1109/IBCAST.2019.8667170>.
- [42] N. H. Sahrir, M. A. M. Basri, "Modelling and Manual Tuning PID Control of Quadcopter," *Control, Instrumentation and Mechatronics: Theory and Practice*, pp. 346-357, 2022, [https://doi.org/10.1007/978-981-19-3923-5\\_30](https://doi.org/10.1007/978-981-19-3923-5_30).
- [43] A. N. Ponce, A. A. Behar, A. O. Hernández, and V. R. Sitar, "Neural Networks for Self-tuning Control Systems," *Acta Polytech.*, vol. 44, no. 1, pp. 49-52, 2004, <https://doi.org/10.14311/514>.
-