



Prediction And Data Collection for Optimized Pid Gains in Stability Control of Quadcopter Using Anfis Model

*Aminu Abdullahi Umar¹, Abubakar Surajo Imam², Muhammad Ahmad Baballe³, Amanatu Kabir⁴, Amina Ibrahim⁵

^{1,2,3} Department of Mechatronics Engineering Nigerian Defence Academy Kaduna Nigeria.

^{4,5} Department of Computer Science, School of Technology, Kano State Polytechnic, Nigeria.

DOI: 10.5281/zenodo.15777902

Submission Date: 21 May 2025 | Published Date: 30 June 2025

*Corresponding author: [Aminu Abdullahi Umar](#)

Department of Mechatronics Engineering Nigerian Defence Academy Kaduna Nigeria.

Abstract

The quest for technological innovation in UAV systems, particularly quadcopters, underscores the critical need for this study. In the contemporary landscape, where UAVs play pivotal roles in defence, surveillance, disaster response and environmental monitoring, the demand for enhanced operational stability and efficiency is ever-increasing. This study presents the development of an adaptive neuro fuzzy inference system with proportional integral derivative controller for an improved stability control of a quadcopter using Webot environment simulation software. The system combines the potential in ANFIS of human-like reasoning of fuzzy systems with the learning capabilities of neural networks and the traditional control systems of PID for offering a powerful tool for controlling complex systems nature of quadcopter. Training data for roll, pitch and yaw were generated using python script to train the ANFIS model for an optimized gain of the PID. The trained model achieved an accuracy of 78.1% with the optimized gains found to be very close to the original gains. The proposed controller showed a very good stability and settling response result compared to the PID controller been used alone. This research contributes to improve the stability of a quadcopter food quality monitoring, inventory management, and storage accountability in refrigeration systems. Future enhancements should focus on integrating image recognition into the user interface to improve accessibility and user experience.

Keywords: Adaptive Neuro Fuzzy Inferencing System (ANFIS), Webot Environment, Proportional Integral Derivative.

I. INTRODUCTION

The advancement of Unmanned Aerial Vehicles (UAVs), particularly quadcopters, has revolutionized a wide range of industrial and defence applications (Gopalakrishnan, 2016). Despite their versatility, maintaining flight stability remains a core challenge in quadcopter deployment. Unlike fixed-wing UAVs, quadcopters possess inherently nonlinear and underactuated dynamics. These characteristics, combined with environmental disturbances such as wind gusts and variable payloads, result in a system highly sensitive to changes in flight conditions (Ma, 2019).

Traditionally, Proportional-Integral-Derivative (PID) controllers have been employed to regulate quadcopter dynamics due to their computational efficiency and relatively simple implementation (Candan, Beke, & Kumbasar, 2018). However, the traditional way of control in them give a chance for researchers to focus in making them adaptive in nature. To address this limitation, research has increasingly focused on intelligent control methods. One such technique is the Adaptive Neuro-Fuzzy Inference System (ANFIS), which combines the learning capability of neural networks with the linguistic reasoning of fuzzy logic (Dorzhigulov, Bissengaliyul, Spencer Jr, Kim, & James, 2018). By making the PID gains an adaptive one by the ANFIS, they become optimized thereby achieving smooth flight and disturbance tolerance in quick response to changes during operation.

II. REVIEW OF LITERATURE

Naima *et al* (2021) proposes a quadcopter control design using optimized PID controllers to control the translational and rotational motions of a quadrotor system with 6 degrees of freedom. Teaching-Learning Based Optimization (TLBO) algorithm was used to obtain the optimal PID controller gains by minimizing the Integral Time Absolute Error (ITAE) criterion. The work shows that combining PID control with TLBO optimization provides a simple but powerful control scheme for quadrotor systems but the error in this work needs to be reduce to attain optimal stability of the quadcopter by the use of more advance control mechanism.

Abdullahi *et al* (2022) proposed a model-based design, HIL testing, and rapid control prototyping of a low-cost POC quadcopter with stability analysis and control. PID and lyapunov stability analysis were used to attain stability. The test was conducted in both an open and confined simulation environment. The system demonstrated a stable response in the confined room, yielding relatively good results, but results collection was challenging due to area limitations in the environment. The average flight time was 9 minutes and 7 seconds, which is quite close to the estimated time a stable flight was achieved using simulation results; the LQR controller displayed an accuracy of 75%; the results obtained in the environment happened to be inaccurate, and the system lacks real-time communication between the system and the control.

Ginting *et al* (2022) proposed a work on a FLC using a geometric control technique to control the quadcopters attitude. The controller's input is the angular velocity and exponential coordinate error, and its output is torque. The mamdani method was used as the foundation for the algorithm, and five membership functions (N, NS, Z, PS, and P) were applied to the inputs and output. Based on the expected attitude for maneuvers, it was discovered that the quadrotor's attitude could be controlled on one or all axes with a steady-state error of roughly 0.02 rad. To achieve an even better optimal result, the fuzzy control needed to be combined with the ANFIS approach.

Fahmizal *et al* (2023) Proposed a drone designed with a "V" shaped frame to optimize the center of mass distribution, the design allows the servomotor torque reaction to be parallel to the rotation axis during pitch movements. ATmega328P microcontroller was used as the main controller, PID controller was used for attitude control and separate PID controllers are implemented for roll, pitch, and yaw control. The result shows a RMSE of 2.3728 was achieved for roll and 4.4219 for pitch.

Framuja *et al* (2024) proposed a work on the design and implementation of roll, pitch, and yaw simulation system for quadrotor control using LQR and PID algorithms. The method employed in the study was achieved through mathematical modeling of the DC motor within the PID controller framework, enabling seamless integration into the LQR calculation. The results demonstrate the comparative advantages of each control method, highlighting the practical implications for applications requiring precise rotational speed control, the values that were captured are 0.000004, 0.000005 and 0.000005 for the angles errors. The adaptability of the control algorithm needs to be improved to a little bit higher level by the ANFIS tool.

III. MATERIALS, EQUIPMENT AND METHODS

The materials used in the development of this work are presented in Table 3.1.

Table 3.1 List of materials and Equipment's

S/N	Name	Description
1	Webot environment software	Material
2	Python script	Material
3	Joystick	Equipment
4	Core i7 computer	Equipment

The methodology employed in this study focuses on developing and validating an algorithm for quadcopter stability control for flying in a disturbance scenario. Utilizing the key features of PID and ANFIS technology helps in achieving that.

Webot Software Environment: A professional open-source robot simulation program is called webots. It enables the modeling, programming, and testing of a variety of robots including quadcopters in physically realistic three-dimensional environments (Dwi, Pebrianti, Syafiq Suhaimi, Bayuaj, & Hossain, 2023).

Python Script: Python is the primary language used to construct the control and simulation system, version 3.7 was chosen for compatibility with contemporary libraries and webots APIs (Gang, et al., 2023).

Joystick: The term "joystick" refers to either radio control (RC) joysticks used in actual drone applications or universal serial bus (USB) game controllers that are frequently found in simulators (contributors, n.d.).

Core i7 Computer: The Core i7 computer's default central processing unit (CPU) running frequency, which normally falls between 2.3 GHz and 3.6 GHz. This establishes the processor's baseline performance under low load. Under high-performance demands, the CPU's maximum speed can exceed 5.0 GHz in certain variants (Eric, 2022).

3.1 BLOCK DIAGRAM OF THE METHODOLOGY

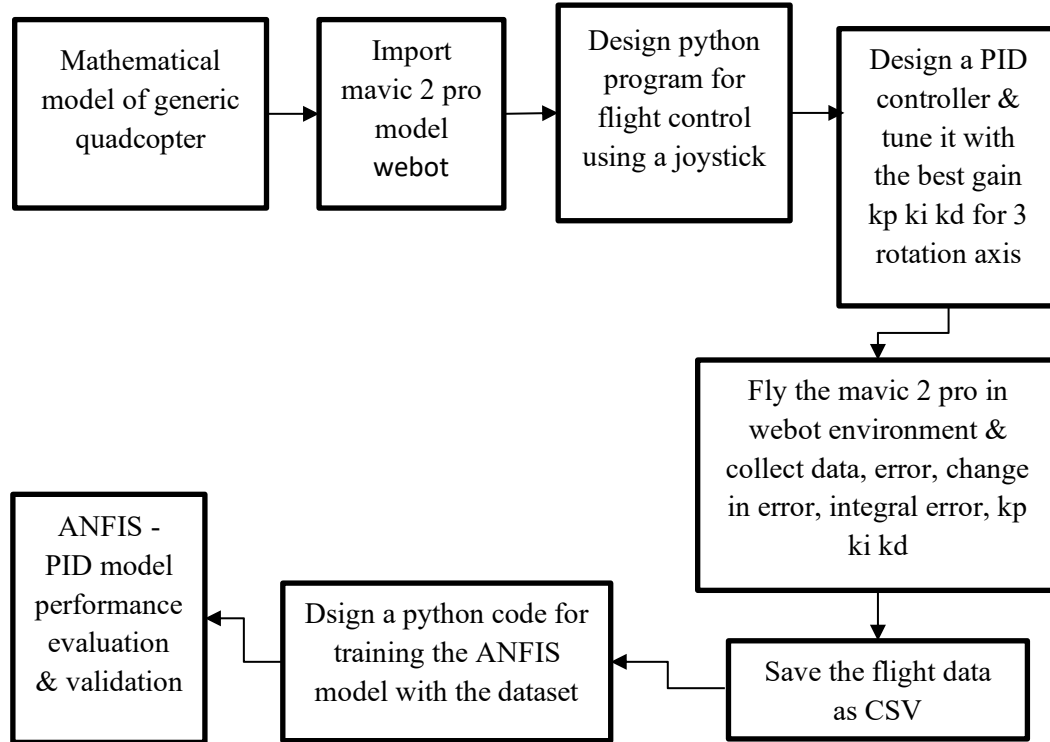


Figure 3.1: Functional block diagram of the methodology

As seen in Fig. 3.1, the development process for the mavic 2 pro drone model involves several key stages. It begins with the mathematical derivation of a generic quadcopter model and specific for the mavic 2 pro drone and importing the mavic 2 pro model into the webot environment. A Python program is then designed to control the drone's flight using a joystick. To enhance stability, a PID controller was developed and tuned with optimized gain values for the three rotational axes. The drone was flown in the webot environment to collect flight data, including error, change_in_error, integral error, and the PID parameters K_P , K_I and K_D , which are saved as a CSV file. This dataset is then used to create a python code for training an ANFIS model, ending in performance evaluation and validation of the ANFIS model. The individual steps are given in the subsequent sub chapters.

3.2 MATHEMATICAL EQUATION

The quadcopter is treated as a rigid body with 6-DOF, and having a combination of translational and rotational motions (Imam, 2014). The quadcopter model was adopted so that it can serve as a basis for designing the control scheme. Equations 3.1-3.12 illustrate the mathematical model.

$$V_B = [u \ v \ w]^T \quad 3.1$$

$$\omega_B = [p \ q \ r]^T \quad 3.2$$

$$\vec{f} = m \cdot \left[\frac{d}{dt} \vec{v} \right]_I \quad 3.3$$

$$Q_B = [h_x \ h_y \ h_z]^T \quad 3.4$$

A quadcopter is controlled by independently varying the angular speed of the four rotors driven by brushless DC electric motors. Each rotor produces a thrust and a torque whose combination generates the vehicle's control input.

$$T_t = T1 + T2 + T3 + T4 \quad 3.5$$

$$R_m = l(T4 - T2) \quad 3.6$$

$$P_m = l(T1 - T3) \quad 3.7$$

$$Y_m = (\tau_2 + \tau_4 - \tau_1 - \tau_3) \quad 3.8$$

Furthermore, the orientation of the quadcopter body must also be compensated during position control. The compensation is attained using the transpose of the rotation matrix in Equation 3.9

$$R_{xyz} = R(x, \varphi)R(y, \theta)R(z, \psi) \quad 3.9$$

They are represented as in the matrixes 3.10, 3.11 and 3.12 (Imam, 2014).

$$R(x, \varphi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \psi & -\sin \psi \\ 0 & \sin \varphi & \cos \varphi \end{bmatrix} \quad 3.10$$

$$R(y, \theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix} \quad 3.11$$

$$R(z, \psi) = \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad 3.12$$

3.3 Model Adaptation in Webots Environment.

The mavic 2 pro webots model is a Unified Robot Description Format (URDF) that gives the links, joints surfaces and sensors of the physical drone. Properties like mass, inertia are factored into the URDF code to match real-world specifications. A snippet of the UEDF code is given in Fig. 3.19.

```
<robot name="mavic2pro">
  <!-- Base Link -->
  <link name="base_link">
    <inertial>
      <mass value="0.907"/>
      <inertia ixx="0.002" ixy="0.0" ixz="0.0" iyy="0.002" iyz="0.0"
      izz="0.004"/>
    </inertial>
    <visual>
      <geometry>
        <box size="0.5 0.5 0.1"/>
      </geometry>
      <material name="grey">
        <color rgba="0.5 0.5 0.5 1.0"/>
      </material>
    </visual>
  </link>
</robot>
```

Figure 3.2: Snippet of the UEDF code

The Fig. 3.3 shows the selection of the DJI mavic drone. The mavic_2_pro.wbt file was selected the world was loaded for simulation as shown in Fig. 3.4.

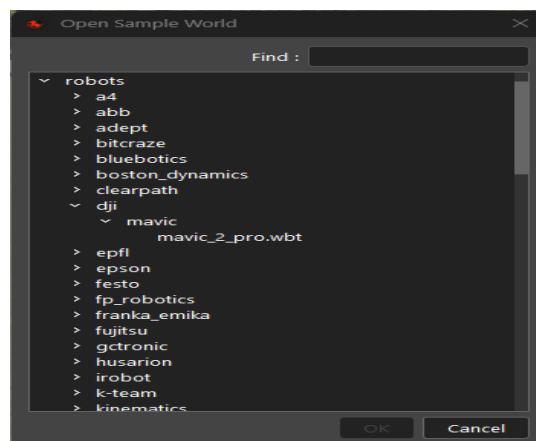


Figure 3.3: Dji mavic 2 adaptation

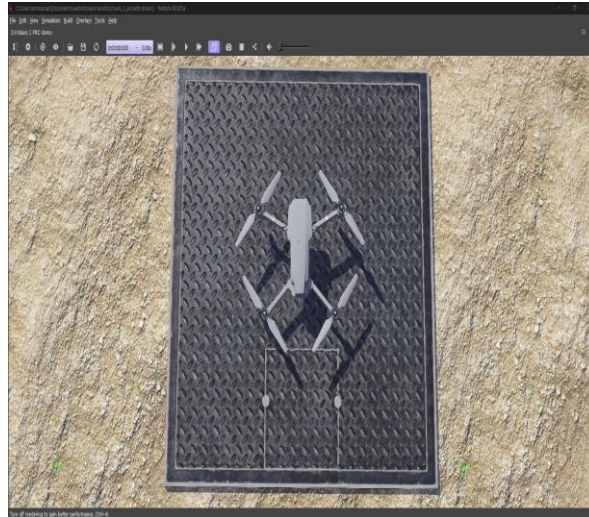


Figure 3.4: Mavic 2 pro drone in Webot.

3.4 PID Control Mechanism

A PID controller works by computing an error value as the difference between the desired setpoint and the actual measured state. The controller then applies corrective control actions based on three terms as shown in Equations 3.13, 3.14 and 3.15.

1. Proportional (P) Term

- The proportional term K_p produces an output that is proportional to the current error. A higher K_p increases the system's response speed but can lead to instability if set too high.
- The proportional control equation is given by 3.13.

$$p = K_p e(t) \quad 3.13$$

- Where $e(t)$ = the tracking error at time t

2. Integral (I) Term

- The integral term K_I accounts for accumulated past errors to eliminate steady-state error.
- The integral control equation is given by 3.14.

$$I = K_I \int_0^t e(\tau) d\tau \quad 3.14$$

- A higher K_I improves accuracy but can lead to overshooting and oscillations.

3. Derivative (D) Term

- The derivative term K_D predicts future errors by calculating the rate of change of the error, helping to dampen oscillations and improve stability.
- The derivative control equation is given by 3.15.

$$D = K_D \frac{de(t)}{dt} \quad 3.15$$

- A higher K_D reduces overshoot

3.5 Tuning the Control System

The ziegler-nichols method was employed to tune the K_p , K_I and K_D gains. This empirical tuning approach is widely used for achieving a balanced trade-off between responsiveness and stability.

3.6 Data Collection for Design of the ANFIS Model

A Python script was developed to systematically collect flight data from the mavic 2 pro drone during its operation, enabling the creation of a comprehensive dataset for tuning the ANFIS model. The Fig. 3.5 shows the flowchart of the process.

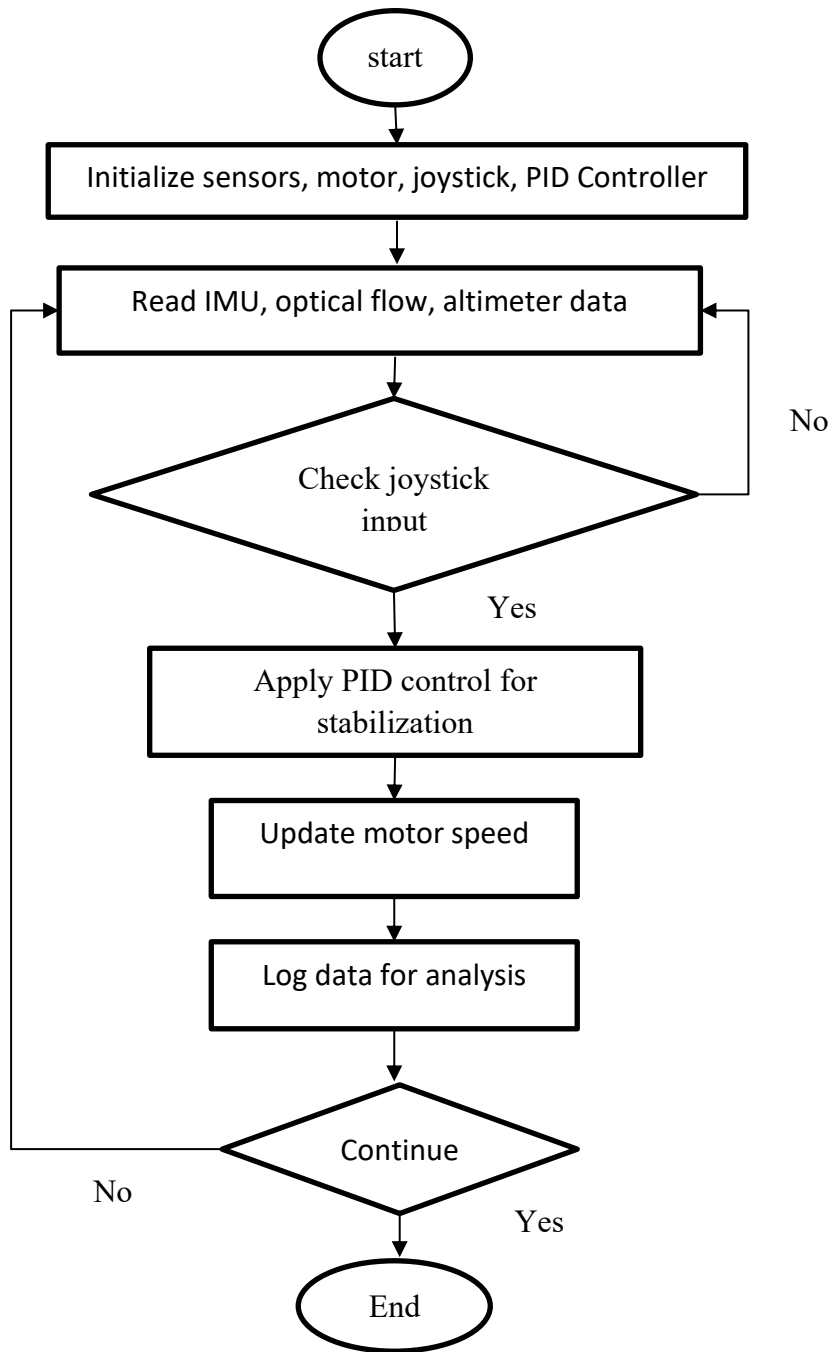


Figure 3.5: Data collection flow chart.

After each of the process for the Roll, Pitch and Yaw training, the data collected were captured and they are presented in Tables 4.1, 4.2 and 4.3 of this work. A tensor flow keras model was designed to optimize the PID gains by implementation of an ANFIS. The Fig. 3.6 shows the steps followed during the training.

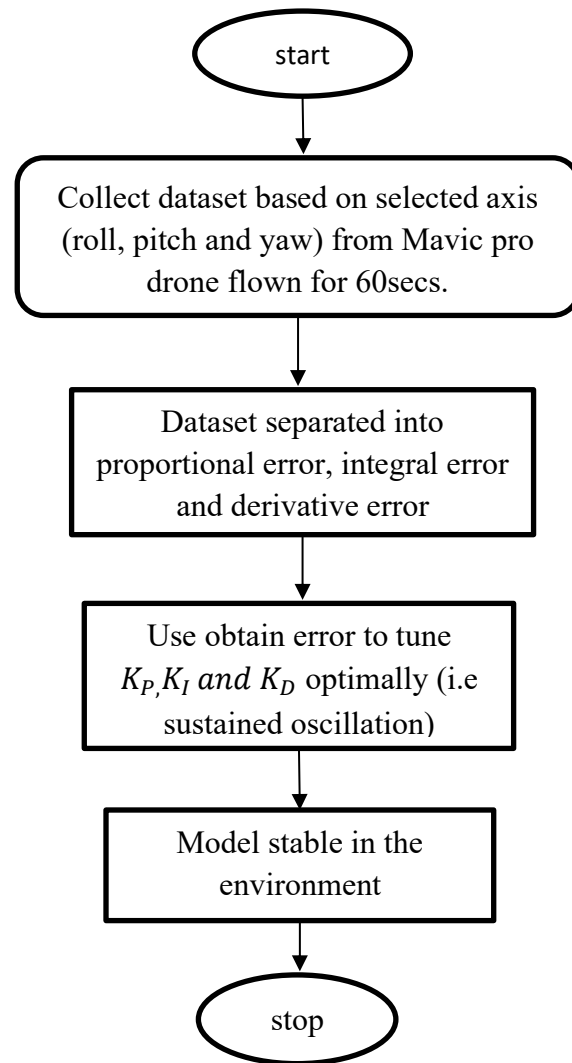


Figure 3.6: ANFIS PID model training steps

3.7 ANFIS Model

The ANFIS is a two input one output system that gives an optimized output with five layers of an intelligent and optimized process. The ANFIS will select the type of rule and membership function for the training of the system. The inputs will be pass in the form of crisp values thereby passing into the fuzzification process and some membership functions are pass over them. The inference engine will choose an appropriate rule in layer three which is in the form of IF- THEN rule for the particular input (Teja & Reddy, 2021).

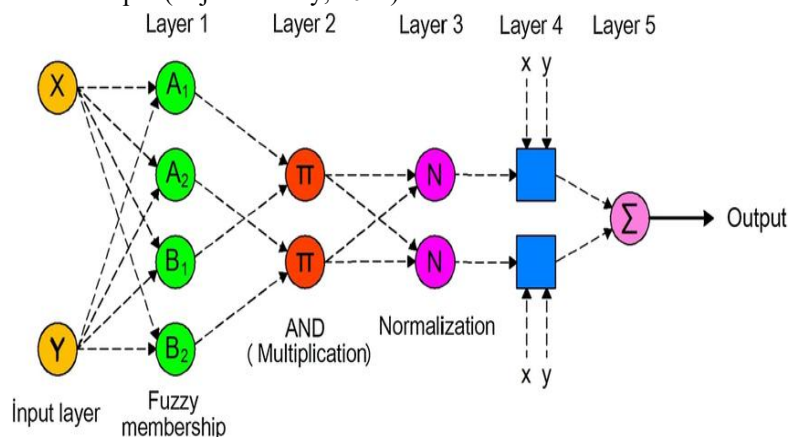


Figure 3.7: Two input one output ANFIS model structure e (Hosseini, M, M, & MA, 2014)

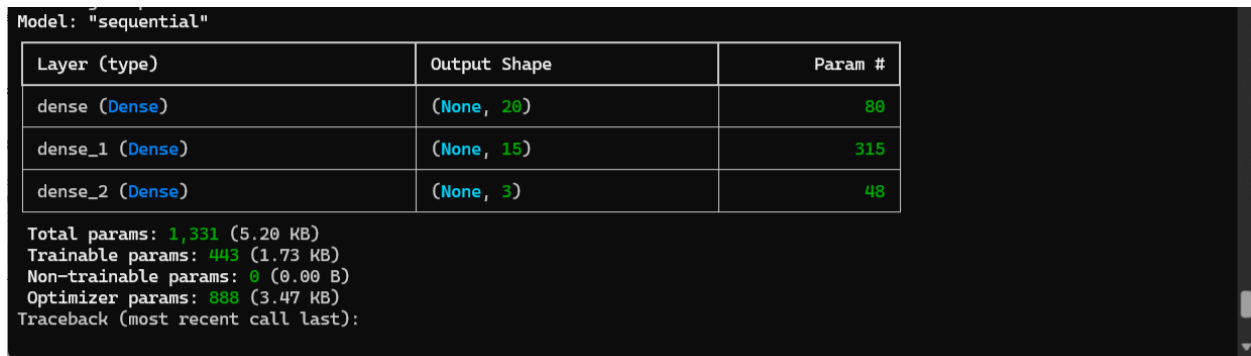
Table 3.2: Training Rules

Rule	Condition	Kp	Ki	Kd	Description
1	Error is large and increasing	High	Low	Medium	Corrects large deviations quickly, prevents overshoot, and dampens oscillations.
2	Error is small and stable	Low	High	Low	Fine adjustments, accumulates changes slowly, and avoids excessive corrections.
3	Error is large and decreasing	Medium	Low	High	Prevents overshooting as the system approaches stability.
4	Change in error is rapid	Low	Low	High	Counteracts sudden changes to prevent overshoot.
5	Integral error is large	Low	High	Low	Minimizes long-term steady-state errors by integrating past errors over time.

Table 3.2 gives the rules employed in the ANFIS model were the are adjusted as either low, medium and high based on a given condition. This will go a long way in making the drone attain a greater stability.

3.8 Neural Network Architecture

The model consists of a sequential neural network with the following layers as shown in Fig.3.9.



Layer (type)	Output Shape	Param #
dense (Dense)	(None, 20)	80
dense_1 (Dense)	(None, 15)	315
dense_2 (Dense)	(None, 3)	48

Total params: 1,331 (5.20 KB)
Trainable params: 443 (1.73 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 888 (3.47 KB)
Traceback (most recent call last):

Figure 3.8: Neural network architecture.

The Fig. 3.8 we have a total parameter of 1331 which were divided into trainable, non-trainable and an optimizer parameter, each having a portion of the overall parameters for processing during the training. During the training, the layers below were generated for the neural network architecture of the K_p , K_i and K_d gains.

1. Input layer:
 - The input layer accepts a 3-dimensional input vector, which represents key control parameters affecting pitch dynamics.
 - The exact inputs depend on the application but may include error signals, rate of change, and external disturbances.
2. Fuzzy membership approximation layer:
 - The first hidden layer is a dense layer with 20 neurons and a tanh activation function.
 - It approximates fuzzy membership functions, which help model nonlinear relationships in the system.
3. Rule layer:
 - The second hidden layer contains 15 neurons, also activated by tanh.
 - This layer represents fuzzy logic rules, capturing complex interactions between inputs and outputs.
4. Output Layer:
 - The final layer has 3 neurons with a linear activation function.
 - These neurons correspond to the optimized PID parameters (K_p , K_i and K_d) needed for roll, pitch and yaw control.

3.9 Model Compilation and Training

- The model was compiled using the Adam optimizer, which is effective for non-convex optimization problems.
- The MSE loss function was used to measure the deviation between the predicted and actual PID parameters.
- The model was trained for 300 epochs with a batch size of 10, allowing it to learn an optimal mapping from input features to PID gains.

The training was done for the roll, pitch and yaw axis and models were created.

IV. RESULTS AND DISCUSSIONS

Results and discussions are presented in this section

4.1 PID Result

Tables 4.1, 4.2 and 4.3 present portion of the training data set for the roll, pitch and yaw respectively which were used for the training of the ANFIS. The features selected to train the ANFIS model are error, change in error and integral Error. The first column in Table 4.1 indicates the error which is the difference between the actual and the desired value, column 2 and 3 indicates the change in error and the integral error, while column 4-6 indicates the values of K_p , K_i and K_d . It is worth noting that different values were recorded in some point to show how the ANFIS model will be train based on different conditions that may warrant adaptation and improvement in the stability of the drone. The values of the K_p , K_i and K_d as shown during the training maintain values that were best suited for the optimization of the K_p , K_i and K_d values.

Table 4.1: Roll Dataset

Error	Change in Error	Integral Error	Kp	Ki	Kd
0.000064	0	0	0.21	0.1	0.1
0.000064	0.000005	0.000001	0.21	0.1	0.1
0.000064	0.000003	0.000001	0.21	0.1	0.1
0.000064	0.000002	0.000002	0.21	0.1	0.1
0.000064	0.000002	0.000002	0.21	0.1	0.1
0.000064	0.000001	0.000003	0.21	0.1	0.1
0.000064	0.000001	0.000003	0.21	0.1	0.1
0.000064	0	0.000004	0.21	0.1	0.1

Table 4.2: Pitch Dataset

Error	Change in Error	Integral Error	Kp	Ki	Kd
-3.992415	0	0	0.21	0.2	0.09
-3.993121	-0.08817	-0.03992	0.21	0.2	0.09
-3.993685	-0.07051	-0.07187	0.21	0.2	0.09
-3.994135	-0.0563	-0.10382	0.21	0.2	0.09
-3.994495	-0.04495	-0.13577	0.21	0.2	0.09
-3.994782	-0.03588	-0.16773	0.21	0.2	0.09
-3.995011	-0.02863	-0.19969	0.21	0.2	0.09
-3.995193	-0.02284	-0.23165	0.21	0.2	0.09

Table 4.3: Yaw Dataset

Error	Change in Error	Integral Error	Kp	Ki	Kd
-0.000673	0.000000	0.000000	0.130000	0.010000	0.100000
-0.000673	0.000000	-0.000007	0.130000	0.010000	0.100000
-0.000673	0.000000	-0.000012	0.130000	0.010000	0.100000
-0.000673	0.000000	-0.000017	0.130000	0.010000	0.100000
-0.000673	0.000000	-0.000023	0.130000	0.010000	0.100000
-0.000673	0.000000	-0.000028	0.130000	0.010000	0.100000
-0.000673	0.000000	-0.000034	0.130000	0.010000	0.100000
-0.000673	0.000000	-0.000039	0.130000	0.010000	0.100000

4.2. Roll Training Result

The Figs. 4.1, 4.2 and 4.4 shows the training of the roll, pitch and yaw angle respectively which the data was captured and being used for the flying of the drone using the ANFIS-PID controller. The training result for the roll, pitch and yaw data show a MAE of 0.0031, MSE of 0.000 and R^2 score of 0.1617, MAE of 0.0001, MSE of 0.0000 and R^2 score of -33009 and MAE of 0.0002, MSE of 0.000 and R^2 score of -8323 for yaw respectively. The MSE and MAE errors were found to be very negligible which shows an improved accuracy in the proposed controller.

```

C:\Windows\System32\cmd.exe
672/672 1s 2ms/step - loss: 2.6605e-07
Epoch 300/300
672/672 1s 2ms/step - loss: 1.2797e-07
WARNING:absl:You are saving your model as an HDF5 file via 'model.save()' or 'keras.saving.save_model(model)'. This file
format is considered legacy. We recommend using instead the native Keras format, e.g. 'model.save('my_model.keras')'
or 'keras.saving.save_model(model, 'my_model.keras')'.
Training Complete. Model Saved!
Model: "sequential"



| Layer (type)    | Output Shape | Param # |
|-----------------|--------------|---------|
| dense (Dense)   | (None, 20)   | 80      |
| dense_1 (Dense) | (None, 15)   | 315     |
| dense_2 (Dense) | (None, 3)    | 48      |



Total params: 1,331 (5.20 KB)
Trainable params: 443 (1.73 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 888 (3.47 KB)
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. 'model.compile_metrics' will be
empty until you train or evaluate the model.
53/53 0s 3ms/step
• Model Performance:
• Mean Squared Error (MSE): 0.0000
• Mean Absolute Error (MAE): 0.0001
• R² Score: -18756612580163486482432.0000

```

Figure 4.1: Training result for roll.

4.3 Pitch Training Result

```

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22621.4169]
(c) Microsoft Corporation. All rights reserved.

C:\Users\emmanuel\Documents\webot\mavic\controllers\mavicDataCollection>code .

C:\Users\emmanuel\Documents\webot\mavic\controllers\mavicDataCollection>python training.py
2025-03-20 12:45:20.747915: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly d
ifferent numerical results due to floating-point round-off errors from different computation orders. To turn them off,
set the environment variable 'TF_ENABLE_ONEDNN_OPTS=0'.
2025-03-20 12:45:25.625467: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly d
ifferent numerical results due to floating-point round-off errors from different computation orders. To turn them off,
set the environment variable 'TF_ENABLE_ONEDNN_OPTS=0'.
Error Change_in_Error Integral_Error Kp Ki Kd
0 0.000064 0.000000 0.000000 0.21 0.1 0.1
1 0.000064 0.000005 0.000001 0.21 0.1 0.1
2 0.000064 0.000003 0.000001 0.21 0.1 0.1
3 0.000064 0.000002 0.000002 0.21 0.1 0.1
4 0.000064 0.000002 0.000002 0.21 0.1 0.1
Training Data Shape: (6719, 3) (6719, 3)
Testing Data Shape: (1680, 3) (1680, 3)
C:\Users\emmanuel\AppData\Roaming\Python\Python312\site-packages\keras\src\layers\core\dense.py:87: UserWarning: Do not
pass an 'input_shape'/'input_dim' argument to a layer. When using Sequential models, prefer using an 'Input(shape)' ob
ject as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
2025-03-20 12:45:46.606571: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to
use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler f
lags.
Epoch 1/300

```

Figure 4.2: Training result for pitch a.

```

Total params: 1,331 (5.20 KB)
Trainable params: 443 (1.73 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 888 (3.47 KB)
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. 'model.compile_metrics' will be
empty until you train or evaluate the model.
53/53 0s 2ms/step
• Model Performance:
• Mean Squared Error (MSE): 0.0000
• Mean Absolute Error (MAE): 0.0031
• R² Score: 0.4022

```

Figure 4.3: Training result for pitch b.

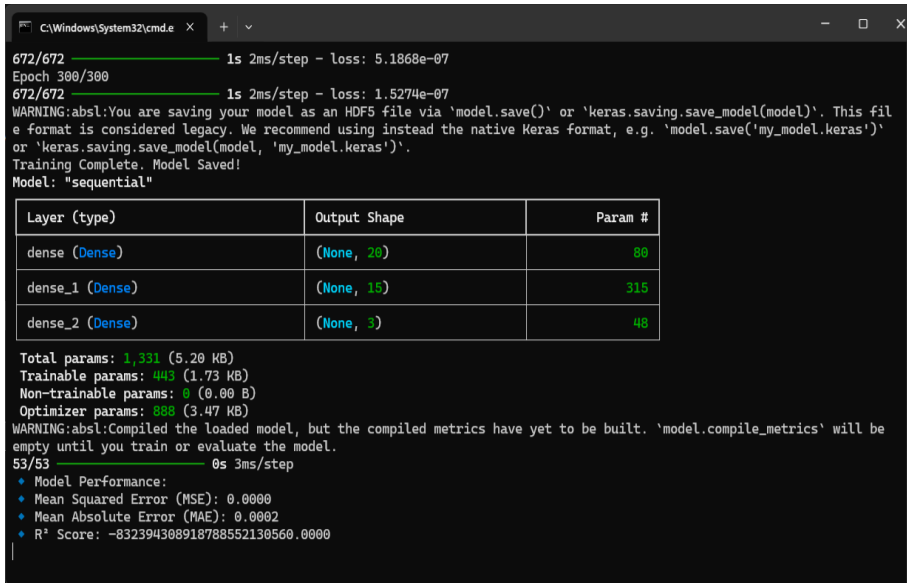


Figure 4.4: Training result for yaw b.

The model for the ANFIS PID model graphs that were generated during the training is as shown in Figs. 4.5, 4.6. and 4.7 for pitch, yaw and roll angles respectively to show the accuracy of the model. The x-axis represents the actual values of the PID gains K_p , K_i and K_d while the y-axis presents the predicted values of those gains by the ANFIS model. The colors blue, green and red represents the proportional, integral and derivatives gains respectively. All three sets of PID gains K_p , K_i and K_d are clustered closely along a diagonal trend, indicating that ANFIS predictions closely match the actual values which reflects a high level of accuracy of the model. For the K_p of pitch, yaw and roll the actual and predicted values range from approximately 0.20 to 0.24, 0.13 to 0.14 and almost the same for roll respectively which followed the actual values quite well with minimal deviation thereby showing a very strong prediction performance for K_p . Likewise, for the K_i and K_d for pitch, yaw and roll the value range are 0.18 to 0.20 and 0.09 to 0.12 and 0.10 to 0.11 and 0.01 to 0.01 and almost the same for roll respectively which shows a reliable prediction.

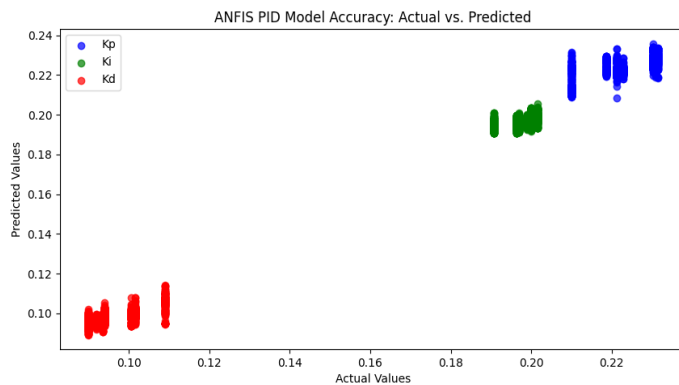


Figure 4.5: ANFIS PID model for pitch.

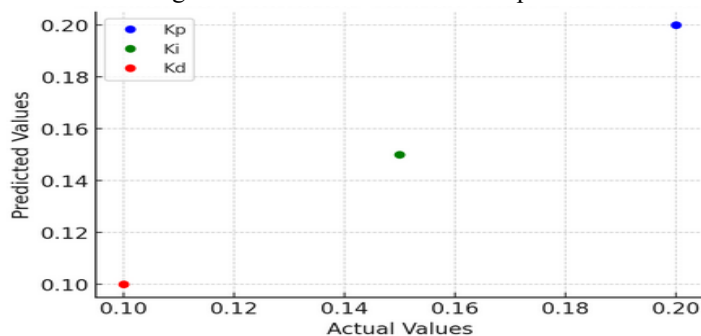


Figure 4.6: ANFIS PID model for roll.

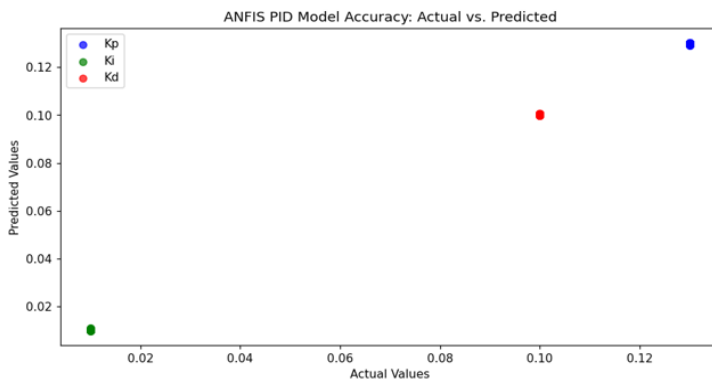


Figure 4.7: ANFIS PID model for yaw.

The three plots demonstrates that the ANFIS model accurately predict PID controller gains, making it a promising method for autonomous control system tuning in this dynamic application of quadcopter stability. The close alignment of predicted vs. actual values shows robustness of the model.

V. CONCLUSION

The ANFIS model was used to predict the roll, pitch and yaw values considering the actual values. The predicted values were found to be very close to the actual value giving a better accuracy. After the training of the ANFIS model, it provides an optimized gain with a very negligible errors of the roll, pitch and yaw angles which gives the quadcopter system ability to be stable while in operation. The PID gains turns out to be adaptable to different operating condition.

REFERENCE

1. Abdullah, Irfan, M., Khan Gufran, A., Arslan Ahmaed, M., Syed Ali, A., Zulfiqar, M., & Adil. (2022). Model-Based Design HIL Testing and Rapid Control Prototyping of a Low Cost POS Quadcopter with Stability Analysis and Control. Article. *Hindawi Complexity*, doi.org/10.1155/2022/1492170.
2. Contributors, W. (n.d.). *Game controller*. Retrieved from Wikipedia, The Free Encyclopedia: https://en.wikipedia.org/wiki/Game_controller. 29/6/2025
3. Dwi, Pebrianti, M., Syafiq Suhaimi, L., Bayuaj, M., & Hossain. (2023). Exploring Micro Aerial Vehicle Mechanism and Controller Design Using Webots Simulation. *Journal of Intelligent Manufacturing & Mechatronics*, Vol. 5, No. 2, pp. 41-49.
4. Eric, . (2022). *File:Intel CPU Core i7 12700K CPU bottom view.jpg*. Retrieved from Wikimedia Commons: https://commons.wikimedia.org/wiki/File:Intel_CPU_Core_i7_12700K_Alder_Lake_bottom.jpg. accessed on 29/6/25
5. Fahmizal, Hanung, Adi Nugroho, D., Imam Cahyadi, A., & Igi. (2023). Attitude Control and Low Cost Design of UAV Bicopte. *Digital Object Identifier*, Vol. 11, pp. 1-12.
6. Framuja, Fortunaviaza, M. A., Habib Ainudin2, N., & Trisna, A. (2024). Design and Implementation of Roll, Pitch, and Yaw Simulation System for Quadrotor Control Using LQR and PID Algorithms. *Journal of Electrical, Marine and Its Application Technology*, Vol. 2, No. 2.
7. Ginting, A. H., Doo, S. Y., Pollo, D. E., Djahi, H. J., & Mauboy, E. R. (2022). Attitude Control of a Quadrotor with Fuzzy Logic . *Journal of Robotics and Control (JRC)*, Vol. 3, No. 3, pp. 101-106.
8. Gang, Gyoo Jin, P., Pal, Y. K., Chung, H., Ku Kang, T. T., Yetayew, S., & Bhakta. (2023). Modelling, control and development of GUI-based simulator for a quadcopter using nonlinear PID control”. *Australian Journal of Electrical and Electronics Engineering*, Vol. 20, No. 4, pp. 387-399.
9. Hosseini, M., Shahbazian, M., & Takassi, M. (2014). The Design of Robust Soft Sensor using Anfis. *Journal of Instrument Technology*, Vol. 2, No. 1, pp. 9-16.
10. Imam, A. S. (2014). *Control and Navigation of a Quadrotor Subject to Wind Disturbance*. Doctoral dissertation. Newcastle (UK): School of Mechanical and Systems Engineering Newcastle University, United Kingdom.
11. Naima, bouhabzaa, k., karaa, m., & hadjilib, l. (2021). PID controllers design for a quadrotor system using teaching learning based optimization. *Wseas Transactions on Applied and Theoretical Mechanics*, Doi: 10.37394/232011.2021.16.10.
12. Teja, K., & Reddy, K. S. (2021). *Adaptive Neuro Fuzzy Inference System*. B.E. project report, Department of Computer Science and Engineering. Chennai, Tamil Nadu, India: Sathyabama Institute of Science and Technology (Deemed to be University), Chennai, Tamil Nadu, India.

CITATION

Aminu A. U., Abubakar S. I., Muhammad A. B, Amanatu K., & Amina I. (2025). Prediction And Data Collection for Optimized Pid Gains in Stability Control of Quadcopter Using Anfis Model. In Global Journal of Research in Engineering & Computer Sciences (Vol. 5, Number 3, pp. 140–151).
<https://doi.org/10.5281/zenodo.15777902>