SSRG International Journal of Electrical and Electronics Engineering
ISSN: 2348-8379/ https://doi.org/10.14445/23488379/IJEEE-V11I2P109

Volume 11 Issue 2, 74-86, February 2024
© 2024 Seventh Sense Research Group®

Original Article

Optimized Fault-Tolerant PID Controller for Quadrotor Stabilization under Actuator Faults

Robert Siame[1*], Peter Kamita Kihato[1,2], George Nyauma Nyakoe[1,2]

[1]*Department of Electrical and Electronics Engineering,*
Pan African University Institute for Basic Sciences, Technology and Innovation (PAUSTI), Juja, Kenya.
[2]*Department of Electrical and Electronic Engineering, Jomo Kenyatta University of Agriculture and Technology, Juja, Kenya.*

[*]*Corresponding Author : robert.siame@yahoo.com*

Received: 19 November 2023 Revised: 11 December 2023 Accepted: 19 January 2024 Published: 16 February 2024

Abstract *- Fault-tolerant control is essential in guaranteeing the stability and reliability of non-linear real-time systems such as Quadrotor UAVs. Controllers based on this approach are capable of estimating and compensating for faults, model uncertainties, and disturbances in a system. While passive fault-tolerant control is widely used, it is not capable of addressing faults that are not predefined in the control setup. In contrast, active fault-tolerant control can detect, estimate, and compensate for any faults in the system. In the recent past, researchers have enhanced this control approach using artificial neural networks and metaheuristic optimization algorithms. This paper uses a Proportional-Integral-Derivative (PID) controller based on Genetic Algorithms (GA) in conjunction with Active Disturbance Rejection Control (ADRC) to provide optimal control gains for the quadrotor and improve fault tolerance. The Quadrotor UAV system was modeled considering actuator loss-of-effectiveness and sine wave disturbances. The system was simulated and analyzed in MATLAB to demonstrate the effectiveness of the proposed approach. Results showed good performance of the controller in handling faults and disturbances, ensuring stability, and continuously optimizing control gains in real time.*

Keywords *- Active Disturbance Rejection Control, Fault-Tolerant Control System, Genetic Algorithm, Proportional-Integral-Derivative (PID) controller, Quadrotor UAV.*

1. Introduction

Modern life has embraced autonomous operations across different sectors, such as aviation, nuclear power, and automotive. Driven by real-time computing for predictable outputs, these mission-critical systems require precise deadlines [2]. Quadrotor UAVs are an example of mission-critical systems that have six degrees of freedom. They offer agility and have simple construction, ideal for compact and lightweight applications. Their minimal requirements make them suitable for military and commercial use despite occasional reliability issues due to their basic design [3, 4].

Ongoing efforts focus on enhancing quadrotor stability through new control methods and design strategies. Figure 1 shows the quadrotor's four-rotor structure [1]. To ensure system stability and reliability amid disturbances, uncertainties, and faults (actuator, sensor, and system), fault-tolerant controllers have been investigated by many researchers [5, 6]. A Fault-Tolerant Control System (FTCS) refers to a system with the capability to tolerate faults. It is divided into passive and active categories. Achieving fault tolerance often involves introducing redundancy, whether in hardware or analytically. However, this approach increases weight, size, and costs for non-linear dynamic systems. Analytical redundancy, in particular, incurs high computational costs. Despite these challenges, the implementation of fault tolerance is crucial, especially for mission-critical systems, where the absence of such measures could lead to severe consequences in the event of a fault. Extensive efforts have delved into fault-tolerance theory, mainly focusing on passive and active fault-tolerant controls [7, 8]. Before developing a fault-tolerant control system, meticulous attention is given to system requirements, analysis, and planning.

Analytical redundancy can sometimes be preferred over hardware redundancy, but this is not a common practice in aviation applications. To provide real-time fault accommodation, a more advantageous strategy combines analytical and direct redundancy methods. It is believed that this potent combination will be necessary for fault-tolerant systems in the future.

Proportional-Integral-Derivative (PID) controllers, known for their simplicity and widespread use, face challenges in handling the increasing complexities of modern systems, including disturbances and faults. PID controllers lack fault estimation, detection, diagnosis, and isolation, making them less suitable for systems with critical performance requirements. In [9], for single-tank level control systems with a leak fault, a Fault-Tolerant Control (FTC) mechanism that makes use of fuzzy logic and PI hybrid passive FTC is suggested. It can handle pre-set challenges well, but it can't handle unanticipated failures well.

For satellite attitude systems with unknown external disturbance and actuator failure, a passive fault-tolerant control is proposed in [10], which demonstrates improved performance over traditional PI controllers. However, its stability under uncertainties and faults undefined in its setup remains a significant drawback.

The limitations of PID controllers in handling complex and non-linear real-time systems have driven the quest for more advanced controllers that not only accommodate faults but also ensure system stability and reliability [8-10]. Active controllers, specifically Active Fault-Tolerant Controllers (AFTC) and Active Disturbance Rejection Control (ADRC), have gained prominence in literature and research [6].

In contrast to PIDs, AFTC systems differ for each fault, relying on Fault Detection and Isolation (FDI) or Fault Detection and Diagnosis (FDD) for fault estimates. AFTCs provide an additional degree of freedom for adjustment in the event of a fault, reducing operating limitations. However, they are vulnerable if the fault diagnostic system malfunctions.

Although Active Fault-Tolerant Controllers (AFTC) are excellent at detecting and diagnosing faults, real-time systems with rapidly changing dynamics may find their processing time to be unsuitable. ADRC stands out as a workable alternative for this. Introduced by Jingqing Han in the 1980s [11], ADRC effectively rejects external disturbances and uncertainties by dynamically estimating and compensating for disturbances in real time using an Extended State Observer (ESO).

In [12], the effectiveness of PID, linear, and Non-Linear Active Disturbance Rejection Control (NLADRC) for a quadrotor is examined. The investigation shows that ADRC is reliable because even when system disturbances are added, the quadrotor maintains its stability. Abadi et al. [13] proposed combining the flatness tracking controller with ADRC to address limitations in an ideal setting, providing a control rule that adjusts for all disturbance effects and optimizes quadrotor performance. In [14], a cascade ADRC strategy with a two-stage Kalman filter is explored for the fault-tolerant control problem of a quadrotor with an actuator fault. The study demonstrates the effectiveness of ADRC during actuator faults, ensuring steady flight even in the presence of failures. However, there is room for improvement, particularly in optimizing control gains to enhance fault tolerance.

Many approaches based on metaheuristic algorithms are used to provide optimization. In [15], fault-tolerant control and fault detection are carried out on a quadrotor UAV by utilizing the Genetic Algorithm in conjunction with backpropagation. When sensor failures are applied, the deployed mechanism proves to be an excellent upgrade over the traditional backpropagation.

Addressing early-stage fault management challenges due to a lack of reliable data, [16] presents an online learning fault-tolerant controller that employs reinforcement learning with a critical action architecture. This innovative approach utilizes particle swarm optimization to rapidly generate a training dataset, accelerating training by approximating the fault-free system's performance index. Demonstrated in successful tests, this technique surpasses traditional Hebb enhancement rules in reinforcement learning.

Several researchers have investigated techniques for optimizing controller parameters in UAV systems. In [17-19], GA is used in tuning the PID controller, which provides optimal gains to the system, proving to be a mighty algorithm. This has led researchers to attempt enhancing PID controllers for intricate systems like quadrotors through optimization techniques.

In [24], authors investigate PID tuning in a quadrotor using various optimization techniques such as Particle Swam Optimization (PSO), Firefly Algorithm (FA), and GA techniques to ensure that optimal PID gains are provided. This study, however, does not take into consideration actuator LOE in the quadrotor. To deal with actuator LOE, it is appropriate to include an active controller that will provide an estimation of the quadrotor parameters in real-time. This is mainly due to active controllers possessing excellent fault management, particularly AFTC and ADRC. An ADRC proves to be a better fit for quadrotors due to its low computational time compared to AFTCs.

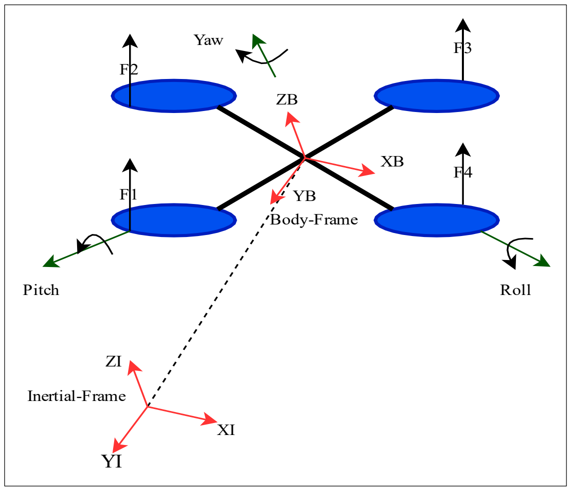

Fig. 1 Quadrotor four-rotor structure [1]

This paper investigates an approach that combines the advantages of both the ADRC and GA-tuned PID controller to mitigate the fault estimation time and continually provide optimized PID controller gains. The proposed control technique ensures that the system can provide stability and reliability even under severe degradation. The hybridized controller is used on a quadrotor UAV to deal with actuator faults as well as external and internal disturbances.

The rest of this paper is organized as follows for the remaining portions: Part 2 covers the formulation of the problem; Part 3 presents the design of the ADRC for fault tolerance and disturbance rejection; Part 4 provides the simulation results and discussion; and Section 5 presents the conclusions.

2. Problem Formulation

2.1. Dynamic Model of the Quadrotor UAV

The quadrotor UAV is modeled under the following assumptions [2]:

a) The framework is inflexible,
b) The structure exhibits symmetry in the axis,
c) The coincidence of the center of gravity and the origin of the body-fixed frame,
d) The propellers are stiff,
e) Thrust and drag are proportional to the square of the propeller's speed.

The quadrotor consists of two frames, that is, Body and Earth, and it is important to establish how the Quadrotor will shift between these two frames. The coordinate frames demonstrate the aerial vehicle's position and orientation with respect to the Earth frame (inertial), as shown in Figure 1.

To transform a rigid body between the two coordinate frames, the rotation matrix R is used, described by the Euler angles φ (Roll), θ (Pitch), and ψ (Yaw). A rotation, as expressed in (1), is defined precisely by rotating the rigid body three times in combination around any three non-planar orientations [20].

$$\begin{cases} R_{x,\varphi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(\varphi) & -s(\varphi) \\ 0 & s(\varphi) & c(\varphi) \end{bmatrix} \\ R_{y,\theta} = \begin{bmatrix} c(\theta) & 0 & s(\theta) \\ 0 & 1 & 0 \\ -s(\theta) & 0 & c(\theta) \end{bmatrix} \\ R_{z,\psi} = \begin{bmatrix} c(\psi) & -s(\psi) & 0 \\ s(\psi) & c(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{cases} \tag{1}$$

Where, $c(*) = \cos(*)$ and $s(*) = \sin(*)$

Using (1), the orthogonal rotation matrix R can be expressed as,

$$R = R_{z,\psi} \times R_{y,\theta} \times R_{x,\varphi} \tag{2}$$

Where, $R^{-1} = R^T$.

With six degrees of freedom, a quadrotor may be characterized by its position (x, y, z) and attitude variables (ψ, θ, ϕ). Its torque is generated in the body frame and is expressed as follows:

$$\tau^B = \begin{bmatrix} lK(-\omega_2^2 + \omega_4^2) \\ lK(-\omega_1^2 + \omega_3^2) \\ b(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{bmatrix} \tag{3}$$

Where, l is the distance from the center of the quadrotor to any of the four propellers, K is the torque coefficient, b is the drag coefficient, and ω is the angular velocity.

Using Euler's equations for rigid bodies, the quadrotor dynamic equation of motion is expressed as;

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \frac{T_B}{m} \begin{bmatrix} s(\psi)s(\varphi) + c(\psi)c(\theta)s(\varphi) \\ s(\psi)s(\theta)c(\varphi) - c(\psi)s(\varphi) \\ c(\theta)c(\varphi) \end{bmatrix} - g \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} - \frac{1}{m} \begin{bmatrix} k_{dx} & 0 & 0 \\ 0 & k_{dy} & 0 \\ 0 & 0 & k_{dz} \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \tag{4}$$

Where; k_{dx}, k_{dy}, and k_{dz} are the drag coefficient, T_B is the total thrust in the body, m denotes the quadrotor's mass of, and g provides the acceleration brought on by gravity.

The non-linear equations of the quadrotor used for controller design (ignoring the drag) are obtained from (4) as:

$$\begin{cases} \ddot{x} = \frac{(s(\psi)s(\varphi) + c(\psi)c(\theta)s(\varphi))u_1}{m} \\ \ddot{y} = \frac{(s(\psi)s(\theta)c(\varphi) - c(\psi)s(\varphi))u_1}{m} \\ \ddot{z} = \frac{(c(\theta)c(\varphi))u_1}{m} - g \end{cases} \tag{5}$$

$$\begin{cases} \ddot{\varphi} = u_2 - lK_1 \dot{\varphi}/I_{xx} \\ \ddot{\theta} = u_3 - lK_2 \dot{\theta}/I_{yy} \\ \ddot{\psi} = u_4 - K_3 \dot{\psi}/I_{zz} \end{cases} \tag{6}$$

Where; u_1 is the input for the thrust control and represents the total thrust force applied to the quadrotor. The quadrotor's roll angle is controlled via the roll control input u_2, which measures the tilt around the x-axis. The pitch control input u_3 determines how far the y-axis is tilted in order to control the quadrotor's pitch angle. Last, input u_4 regulates the quadrotor's yaw rate, which dictates how quickly the yaw angle rotates around the z-axis. (K_1, K_2, K_3) are related to the dynamics and control of the quadrotor and (I_{xx}, I_{yy}, I_{zz}) are inertia moments about the $x, y,$ and z axes, respectively.

Inputs to the controller are then given as;

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} K\sum_{i=1}^{4}\omega_i^2 \\ lK(-\omega_2^2 + \omega_4^2) \\ lK(-\omega_1^2 + \omega_3^2 \\ b(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2 \end{bmatrix} \quad (7)$$

2.2. GA-Based Tuning of PID Controller Gains

PID controller is expressed as demonstrated in Equation (8),

$$u(t) = K_p\, e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt} \quad (8)$$

To get the required control performance for the quadrotor, the gains of the PID controller are adjusted.

To adjust the PID gains, GA is used so that there is a minimal error margin $(e(t))$ between the set value and the actual value. Figure 2 depicts the flowchart of GA, which is modeled after natural selection and uses mechanisms such as crossover, mutation, and selection to repeatedly develop a population of possible solutions toward an ideal solution for a given issue.

To achieve optimal PID tuning gains, the following error criteria are used: Integrated Time Absolute Error (ITAE), Integrated Absolute Error (IAE), and Integrated Square Error (ISE). Equations (9), (10), and (11), respectively, define these. Then, using a Genetic Algorithm, the weighted combination of ISE, IAE, and ITAE is used as an objective function to get PID parameters [21].

$$ISE = \int_{\infty}^{\infty}[e(t)]^2 dt \quad (9)$$

$$IAE = \int_{\infty}^{\infty}|e(t)|dt \quad (10)$$

$$ITAE = \int_{\infty}^{\infty} t|e(t)|dt \quad (11)$$

Therefore, the fitness function for the genetic algorithm can be provided as a combination of the weighted sum of the above three equations as demonstrated below:

$$J(K_c, T_I, T_d) = \omega_1 ISE + \omega_2 IAE + \omega_3 ITAE \quad (12)$$

The weights, ω_1, ω_2 and ω_3, are essential for assigning priority to the fitness function during optimization. Once the optimal gains are converged by the genetic algorithm, the PID control is updated in order to fit in these optimized gains. The tuned parameters are then used in Equation (8).

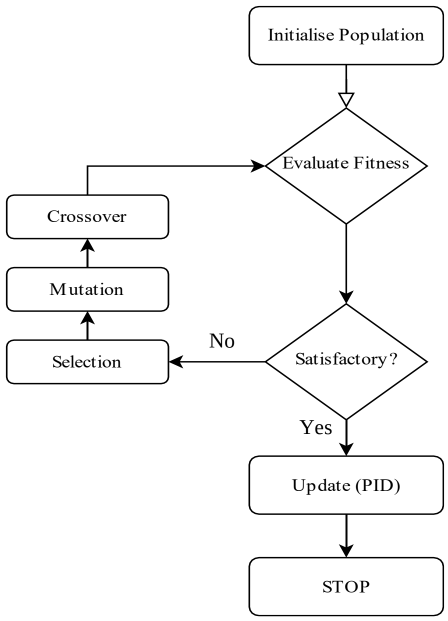

Fig. 2 GA flowchart for tuning PID controller

3. Model Formulation of ADRC for Fault Tolerance and Disturbance Rejection

The ADRC control approach aims to improve system performance when faced with uncertainties and disturbances. It consists of two major parts: a control rule for fault/disturbance compensation and an extended state observer for disturbance/fault estimation.

ADRC is distinguished by its resilience, simplicity, and adaptability, providing efficient performance even with little system model knowledge. This method is beneficial for systems with complicated or ambiguous models since it offers better control and resilience to outside influences. The actuator problems the system will encounter must first be mathematically represented for the ADRC to identify and correct them.

3.1. Actuator Loss of Effectiveness

To mimic actuator loss of effectiveness, the quadrotor's dynamics are further reduced in the manner shown in Equation (13) when simulating actuator faults [22].

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} K & K & K & K \\ 0 & -lK & 0 & lK \\ -lK & 0 & lK & 0 \\ lb & -lb & lb & -lb \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} := A_h \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \quad (13)$$

Where, A_h is the matrix coefficient during the quadrotor's hovering.

Using [23] to simulate the actuator's LOE, where the effectiveness of the actuator will decline in the event of a fault.

The LOE factor for the four actuators is denoted by l_i, i = 1, 2, 3, and 4. Equation (13) should therefore be modified to:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} K & K & K & K \\ 0 & -lK & 0 & lK \\ -lK & 0 & lK & 0 \\ lb & -lb & lb & -lb \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \begin{bmatrix} 1-l_1 \\ 1-l_2 \\ 1-l_3 \\ 1-l_4 \end{bmatrix} \quad (14)$$

This form only works when the quadrotor is in hover to make sure that the system is still functional even when K varies the definition in [23] is used. K_i is not constant while the quadrotor is not hovering, therefore the new LOE is expressed as follows:

$$K_i = \rho\sigma a A r^2 \left[\left(\frac{1}{6} + \frac{\mu_i^2}{4}\right)\theta_0 - (1+\mu_i^2)\frac{\theta_{tw}}{8} - \frac{\lambda_i}{4}\right] \quad (15)$$

Where, θ_0 and θ_{tw} represent the pitching of incidence and twist pitch, respectively, σ and a are the solidity ratio as well as lift slope, A is the reference area, r is the propeller's radius, and ρ is the air density. Furthermore, the rotor's inflow and advance ratios, represented by λ_i and μ_i, are as follows:

$$\lambda_i = \frac{v_1 - \omega}{\omega_i R} \quad (16)$$

$$\mu_i = \frac{V}{\omega_i R} \quad (17)$$

Where;

$$V = \sqrt{\dot{x}^2 + \dot{y}^2} \quad (18)$$

$$v_1 = \sqrt{-\frac{V^2}{2} + \sqrt{\left(\frac{V^2}{2}\right)^2 + \left(\frac{mg}{8\rho A}\right)^2}} \quad (19)$$

Where, V is the horizontal velocity and v_1 is the inflow velocity. Consequently, $K_i \neq K$ when the quadrotor is not in hover, as shown by the following ratio:

$$\eta_i = \frac{K_i}{K} = 1 + \frac{\frac{\theta_0 \mu_i^2}{4} - \frac{\theta_{tw}\mu_i^2}{8} - \frac{\lambda_i}{4} + \frac{\lambda_h}{4}}{\frac{\theta_0}{6} - \frac{\theta_{tw}}{8} - \frac{\lambda_h}{4}} \quad (20)$$

Where;

$$\lambda_h = \frac{\sqrt{\frac{mg}{8\rho A}}}{\omega_h r} \quad (21)$$

With ω_h being the rotational speed during hover. The ratio η_i can be simplified as shown in Equation (22) [23]:

$$\eta_i = a_0 + a_1\lambda_i + a_2\mu_i^2 \quad (22)$$

When utilizing a different quadrotor, it is necessary to identify a_0, a_1 and a_2. In this paper: $a_0 = 1.40$, $a_1 = -5.94$ and $a_2 = 1.42$ will be used. An actuator loses effectiveness when it fails.

Since it is believed that the rotor's drag torque is proportionate to its thrust, Equation (23) can be used to compute the LOE factors in the following manner:

$$\begin{bmatrix} l_1 \\ l_2 \\ l_3 \\ l_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} - A^{-1} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \quad (23)$$

In which A is provided as follows:

$$A = A_h.\ \mathrm{diag}\,([\eta_i\omega_1^2, \eta_i\omega_2^2, \eta_i\omega_3^2, \eta_i\omega_4^2]) \quad (24)$$

For ease in simulation, Equation (24) can be simplified as follows:

$$L_A = \{I_{4\times4} - (\mathrm{diag}\,[l_1, l_2, l_3, l_4])\} \quad (25)$$

Where, $I_{4\times4}$ is a 4×4 identity matrix: When there are partial actuator faults, it can be observed that $l_i \neq 1$, with $i = 1, 2, 3,$ and 4, respectively, suggesting there is a degradation in the performance of the actuator. Otherwise, $l_i = 1$, when the actuator is experiencing no partial faults.

3.2. Quadrotor State-Space Representation

With the quadrotor and actuator fault modeled, the quadrotor mathematical model is converted into a state-space model.

This represents a more controllable and comprehensible framework for describing the behaviour of a quadrotor when developing the ADRC. The state-space is defined as follows:

$$\begin{cases} x = [x, y, z, \dot{x}, \dot{y}, \dot{z}, \varphi, \theta, \psi, \dot{\varphi}, \dot{\theta}, \dot{\psi}]^T \\ u = [u_1, u_2, u_3, u_4]^T \\ y = [x, y, z, \varphi, \theta, \psi]^T \\ L_A = [L_{Ax}, L_{Ay}, L_{Az}, L_{A\varphi}, L_{A\theta}, L_{A\psi}]^T \end{cases} \quad (26)$$

Where by the quadrotor's fault-free state-space model will be provided as follows:

$$\begin{cases} \dot{x} = Ax(t) + g\big(x(t), u(t)\big) + Bu(t) \\ y = Cx(t) \end{cases} \quad (27)$$

The matrices A, B, and C are system matrices of appropriate dimensions, x represents the system state, y is the vector output, while u means the quadrotor's control input.

Finally, g(x(t),u(t)) provides the disturbance, nonlinearities, and model uncertainties within the system. The key sources of uncertainty are the unknown drag coefficients $[K_1, K_2, K_3]$ and the undetermined moment of inertia $[I_{xx}, I_{yy}, I_{zz}]$. Considering a system with actuators faults, Equation (27) can be rewritten as,

$$\begin{cases} \dot{x} = Ax(t) + g\big(x(t), u(t)\big) + Bu_f(t) \\ \quad y = Cx \end{cases} \tag{28}$$

Where,

$$u_f(t) = u(t) + HL_A(t) \tag{29}$$

H represents the actuator fault distribution matrix; multiplicative faults are considered in this research.

3.3. Model of Active Disturbance Rejection Control for Fault-Tolerance

The working principle of ADRC is demonstrated in [11] [12]. The Extended State Observer (ESO) and Tracking Differentiator (TD) are the two main parts of ADRC. Although TD is not discussed much in this study. The state space in Equation (28) can be redefined as;

$$\ddot{y}(t) = f(t) + b_o u(t) \tag{30}$$

Where, the total disturbance, faults included, is given by $f(t) = -a_1 y(t) - cy(t) + (b_u - b_o)u(t) + \Delta$ with Δ representing the external faults (actuator faults and other disturbances), b_o is a nominal measurement of b_u and $a, b, c,$ are unknown plant variables, u is the input, and y represents the output. With the help of an ESO, $f(t)$ is estimated, the resulting control law is written as follows:

$$u(t) = \frac{u_0(t) - z_3}{b_o} \tag{31}$$

Where, z_3 is the estimated value of $f(t)$ by the ESO. Replacing (31) with (30), the result is as follows:

$$\ddot{y}(t) = u_0(t) - z_3 + f(t) \tag{32}$$

Considering that after the estimation of z_3, $f(t) \approx z_3$ which implies that:

$$\ddot{y}(t) = u_0(t) \tag{33}$$

Meaning the ESO has estimated the error. The GA-based PID controller can further control the double integration.

Thereby ensuring that the plant tolerates the faults in the system. Based on Equation (28), ESO is provided by the following equations:

$$\begin{cases} \dot{z}_1 = z_2 - \beta_1(z_1 - \bar{y}) \\ \dot{z}_2 = z_3 - \beta_2(z_1 - \bar{y}) + b_o u(t) \\ \dot{z}_3 = -\beta_3(z_1 - \bar{y}) \end{cases} \tag{34}$$

In the ESO, the estimated states are denoted as z_1, z_2, and z_3, and they correspond to estimated values of x_1, x_2, and x_3 respectively.

With $\beta_i = (1,2,3)$ representing the feedback gain of the ESO. The ESO's characteristic equation is provided by:

$$\lambda = s^3 + \beta_1 s^2 + \beta_2 s + \beta_3 \tag{35}$$

In order to get suitable observer gains, poles are placed in the closed loop, which is at $-\omega_o$, which results in Equation (35) being modified to $\lambda_o(s) = (s + \omega_o)^3$ [14]. Therefore, three ESO feedback gains are obtained:

$$\begin{cases} \beta_1 = -3\omega_o \\ \beta_2 = -3\omega_o{}^2 \\ \beta_2 = -3\omega_o{}^3 \end{cases} \tag{36}$$

The effectiveness of the designed controller is tested under different Quadrotor scenarios. The ESO bandwidth, denoted by ω_o, is usually maintained five to ten times greater than the close-loop bandwidth, ω_c.

Figure 3 shows the general block diagram of the suggested quadrotor control system.

Table 1. Actuator LOE of each motor

Channels	Actuator LOE	
	Motor	**Percentage**
1	Motor 1	0.5
2	Motor 2	0.9
2	Motor 3	0.6
4	Motor 4	0.9

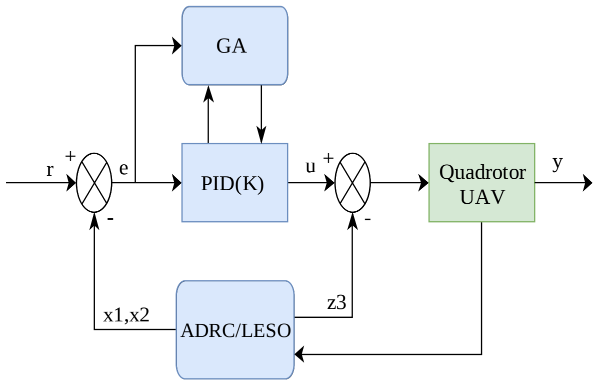

Fig. 3 Overall design of the proposed controller and quadrotor UAV system

4. Simulation Results and Discussion

The following Quadrotor parameters are considered for simulation on MATLAB.

Table 2. Denotes the various parameters of the modeled quadrotor

Parameters	Quadrotor Parameters	
	Units	**Magnitude**
Mass	(kg)	0.5
Gravity	(m/s^2)	9.81
Inertia (I_{xx}, I_{yy}, I_{zz})	(kg/m^2)	$diag\ (5\times10^{-3}, 5\times10^{-3}, 10\times10^{-3})$
Arm Length	(m)	0.25
Thrust Coefficient		3×10^{-6}
Torque Coefficient		1×10^{-7}
Motor Delay		20×10^{-3}
Drag Coefficient		0.25

Table 3. Controller parameters used in the simulation

Channels	Controller Parameters	
	GA-Based PID	**ADRC**
Altitude	$\mathrm{K_d} = 13.4133, \mathrm{K_d} = 87.7034$	$\omega_c = 8, \omega_o = 8\times12, \mathrm{b_A} = \frac{1}{\mathrm{m}}$
Roll	$\mathrm{K_d} = 34.7622, \mathrm{K_d} = 251.1592$	$\omega_c = 12, \omega_o = 12\times25, \mathrm{b_R} = \frac{1}{\mathrm{I_{xx}}}$
Pitch	$\mathrm{K_d} = 34.7622, \mathrm{K_d} = 251.1592$	$\omega_c = 12, \omega_o = 12\times25, \mathrm{b_P} = \frac{1}{\mathrm{I_{yy}}}$
Yaw	$\mathrm{K_d} = 22.6526, \mathrm{K_d} = 274.0315$	$\omega_c = 12, \omega_o = 12\times25, \mathrm{b_Y} = \frac{1}{\mathrm{I_{ZZ}}}$

A controller hybrid between GA-tuned PID and an active controller with tfhe parameters provided in Table 2 is implemented. The control channels for pitch and roll are considered symmetrical in this simulation; hence, the GA-based PID gains for pitch are used in the Roll control channel. The GA parameters were set at generations, 10 with population, and 25 for all the channels.

4.1. Nominal Performance Analysis

To assess the efficacy of the proposed control scheme, a perfect simulation of the quadrotor and its controller was first conducted. This meant that neither external nor internal disturbances or faults were factored into the simulation, as shown in Figures 4, 5, and 7.

4.1.1. Path Tracking

The quadrotor and controller performed as anticipated, with the quadrotor successfully following the reference input with minimal errors, as demonstrated in Figure 4. GA proved an excellent methodology to eliminate the model uncertainties in the system due to the optimized gains. Hence, providing excellent trajectory tracking results.

4.1.2. Motor's RPMs

Under fault-free circumstances, the motors' RPMs are found to be satisfactory, with only slight deviations observed, primarily due to modeling uncertainties in the system, as shown in Figure 7.

4.1.3. Quadrotor's Position

The controller's position remains unaffected as the developed controller accurately follows the referenced trajectory. Similar to the Path Tracking and motor RPMs, only minor deviations are observed in the position, but there are no significant concerns; this is demonstrated in Figure 5.

4.1.4. Euler Angles

The outcomes indicate that the Euler angles (Pitch, Roll, and Yaw) function outstandingly under fault-free conditions, attributable primarily to the developed controller. The GA's fine-tuning of the PID controller is essential to getting the optimum gains. This effectively highlighted the efficacy of the combined GA-based PID and ADRC controller; this is demonstrated by the trajectory reference in Figure 4.

4.2. Fault Tolerance Performance Analysis

To assess the controller's functionality in the event of actuator faults, Table 3 outlines the specific actuator parameters used to simulate a reduction in the actuator. The controller results are provided in Figures 6, 7, 8 and 9.

4.2.1. Path Tracking

To showcase the proficiency of the quadrotor's controller, a specific path was set for the system to adhere to during the Loss of Effectiveness (LOE) scenarios. The controller not only tolerates actuator faults during these periods but also effectively rejects and compensates for these faults. Enhanced by the optimization of the estimated values generated by the ADRC, the controller demonstrates its ability to follow the designated trajectory, as depicted in Figure 9.

4.2.2. Motor's RPMs

To gauge the proposed controller's fault tolerance, a partial actuator LOE was purposefully applied to the actuators. Following this, the controller was adjusted using the Genetic Algorithm (GA) to maintain the Quadrotor's stability and reliability during faults. These faults were introduced at various intervals to observe the controller's response to abrupt and varying degrees of faults. This is depicted in Figure 7.

4.2.3. Quadrotor's Position

Post-actuator LOE, the quadrotor's actual positions in the x, y, and z coordinates remarkably followed the set reference inputs. Notably, the only significant deviation was in the z position. However, this deviation was significantly reduced after multiple rounds of PID tuning using the Genetic Algorithm. Figure 6 illustrates this improvement, serving as evidence of the fault tolerance capabilities of the quadrotor's controller.

4.2.4. Euler Angles

Actuators are purposely subjected to actuator partial LOE in order to assess the proposed controller's fault tolerance capabilities. Following this, the controller was adjusted using the Genetic Algorithm (GA) to maintain the Quadrotor's stability and reliability during faults. These faults were introduced at various intervals to observe the controller's response to abrupt and varying degrees of faults. This is depicted in Figure 7.

4.3. Disturbance Rejection Performance Analysis

The simulation experiment incorporates external disturbances in order to evaluate the efficacy of the suggested technique in more detail. As a controller based on disturbance observation and rejection, the ADRC, which is renowned for its remarkable disturbance rejection abilities, is tested for this particular feature.

The controller's ability to counteract disturbances is evaluated by introducing sine wave disturbances with various characteristics into both the altitude and attitude control channels. The sine disturbance is defined mathematically as $d(t) = A sin(\omega t + \phi)$, where A is the amplitude, ω the frequency, and ϕ the phase angle of the sine wave, as referenced in [15]. Table 4 outlines the specific sine disturbances to be introduced in the simulation at the 25th second. The results are shown in Figures 6, 7, 8, 9, and 11. The sine disturbances were introduced to the system to test the rejection of disturbances (internal or external).

Table 4. Sine wave disturbances are introduced to the system

Channels	Sine Disturbances		
	A	$\boldsymbol{\omega(rad/s)}$	$\boldsymbol{\phi(rad)}$
T	0.7	2	0
τ_φ	0.1	1	$\frac{\pi}{2}$
τ_θ	0.1	1	π
τ_ψ	0.1	1	2π

4.3.1. Motor's RPMs

The RPMs of the motors were examined, and the analysis of the output revealed the impact of the sine disturbances, evident in the RPM plots. Figure 7 displays the motor RPMs that experienced sine disturbances at a specific moment.

4.3.2. Euler Angles

The alteration in motor behaviour is also observed in the dynamics of the quadrotor, particularly in the Euler angles (Pitch, Roll, and Yaw). Figure 8 illustrates that the significant impact of the sine disturbances is predominantly noticeable in all the Euler angles (Pitch, Roll, and Yaw) at the 25th second.

4.3.3. Quadrotor's Position

Following the disturbance in the system, it was observed that the actual positions (x, y, z) closely followed the reference inputs with remarkable accuracy. The only noticeable deviations occurred in the z position. Figure 6 illustrates how these deviations were considerably decreased through PID tuning with the Genetic Algorithm, precisely as faults were introduced.

4.3.4. Path Tracking

To showcase the proficiency of the quadrotor's controller, a path track was set up for the system to follow during sine disturbance scenarios. The controller is adept not only at estimating the disturbances but also at rejecting and compensating for these faults. Enhanced optimization of the values estimated by the ADRC further illustrates the controller's ability to track the designated trajectory, as shown in Figure 9.

The simulation results under various conditions have effectively demonstrated the efficiency of the control scheme.

To reinforce this, the controller was tested in scenarios involving actuator LOE and sine disturbances, as shown in Figures 6, 7, 8, 9, and 11. These clearly indicate the controller's capacity to both tolerate faults and counteract disturbances in the system. This was achieved through the exceptional performance of the ADRC, working in tandem with the GA-based PID controller. The finely tuned PID controller ensured the continuous provision of optimal gains, both in faulty situations and amidst external and internal disturbances.

In this study, the authors were able to achieve improved performance compared to the most recent study [14], mainly due to the addition of a genetic algorithm to ensure optimal gains are obtained. The proposed control scheme provided better tracking of the referenced input by the controller. The quadrotor remained stable even when subjected to two actuators with partial loss of effectiveness (of at least 50%).

The enhancement in addressing actuator faults and disturbances can be attributed to the utilization of separate loops for the ADRC and the GA-tuned PID controller in this study. This segregation ensures that any computational delays incurred by the GA-based PID controller do not impede the responsiveness of the ADRC. Furthermore, as shown in Figure 9, the quadrotor system remained stable and reliable during unpredictable Euler angle rate fluctuations, as illustrated in Figure 11. Which further reinforced the efficacy of the controller design.

4.4. Genetic Algorithm Optimised Gains

A Genetic Algorithm is used to modify the gain settings of the PID (PD) controllers for each of the four channels in order to optimize them efficiently: Altitude, yaw, and pitch (roll). These parameters are detailed in Table 5, including the maximum and minimum limits for the gains. The inclusion of these bounds in the simulations was crucial to achieve faster convergence, a result obtained by experimenting through trial and error. The figure for PID-optimized control gains is shown in Figure 10.

Table 5. Genetic Algorithm optimized gains for PID

Channel	GA Parameters			
	Lower Bound	Upper Bound	Kd	Kd
Altitude	[0,0]	[100,100]	9.96	97.24
Roll	[0,0]	[300,300]	34.76	251.16
Pitch	[0,0]	[300,300]	34.76	251.16
Yaw	[0,0]	[300,300]	22.65	274.03

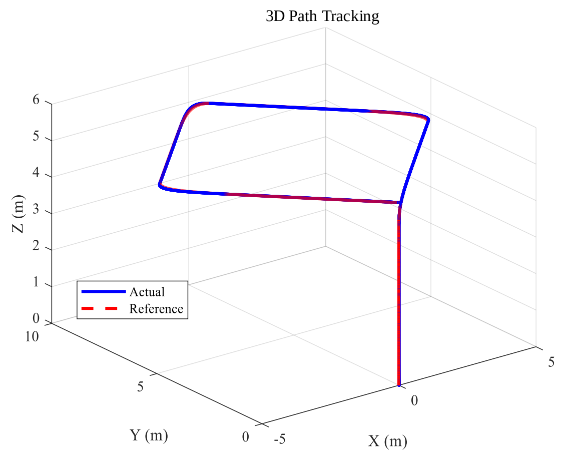

Fig. 4 Motor's RPMs for both the LOE and sine disturbance at 20th and 25th seconds, respectively

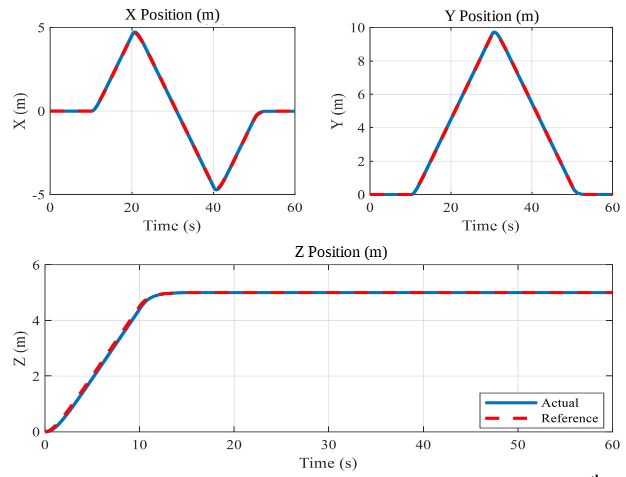

Fig. 5 Motor's RPMs for both the LOE and sine disturbance at 20th and 25th seconds, respectively

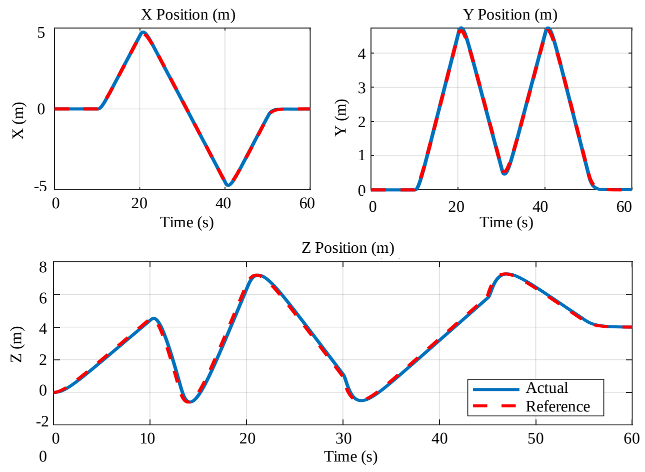

Fig. 6 the quadrotor's position during LOE conditions and sine disturbances, which are injected at the 20th second

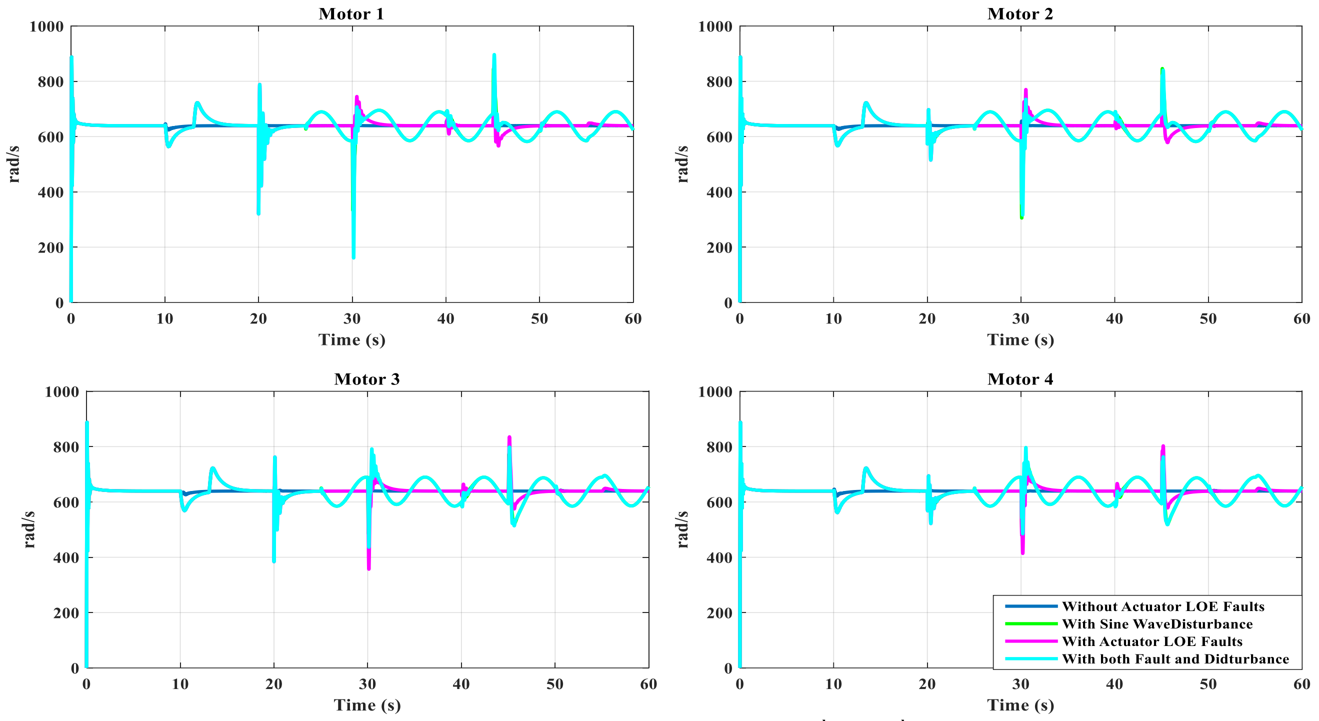

Fig. 7 Motor's RPMs for both the LOE and sine disturbance at 20th and 25th seconds, respectively

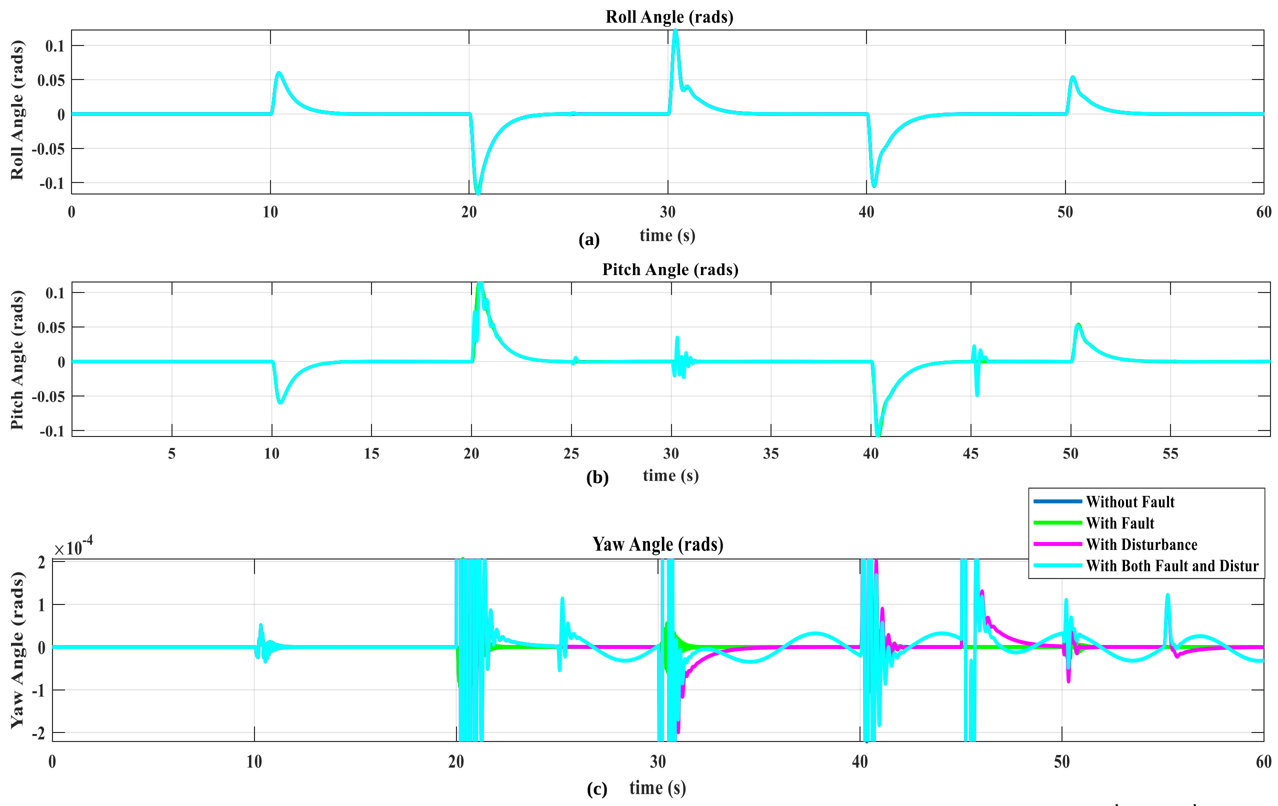

Fig. 8 Motor's Euler angle ((a) Roll, (B) Pitch, and (C) Yaw) plots in the presence of both the LOE and sine disturbance at the 20th and 25th seconds, respectively

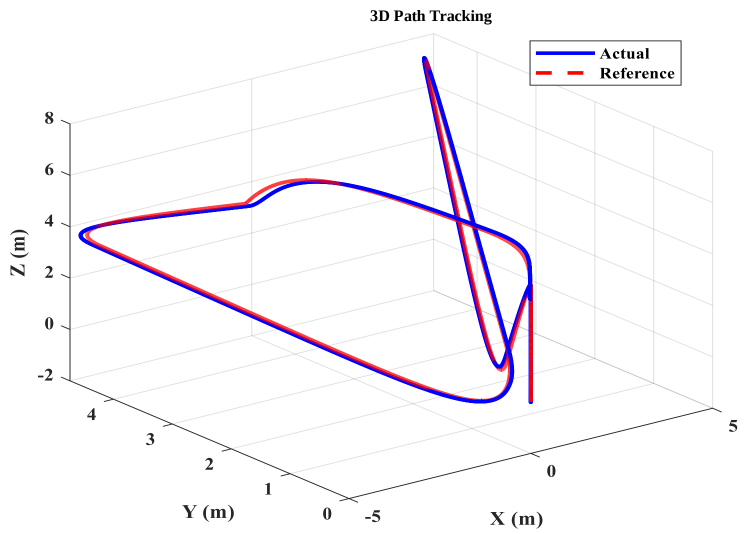

Fig. 9 Trajectory tracking for the quadrotor in both LOE conditions and sine disturbances

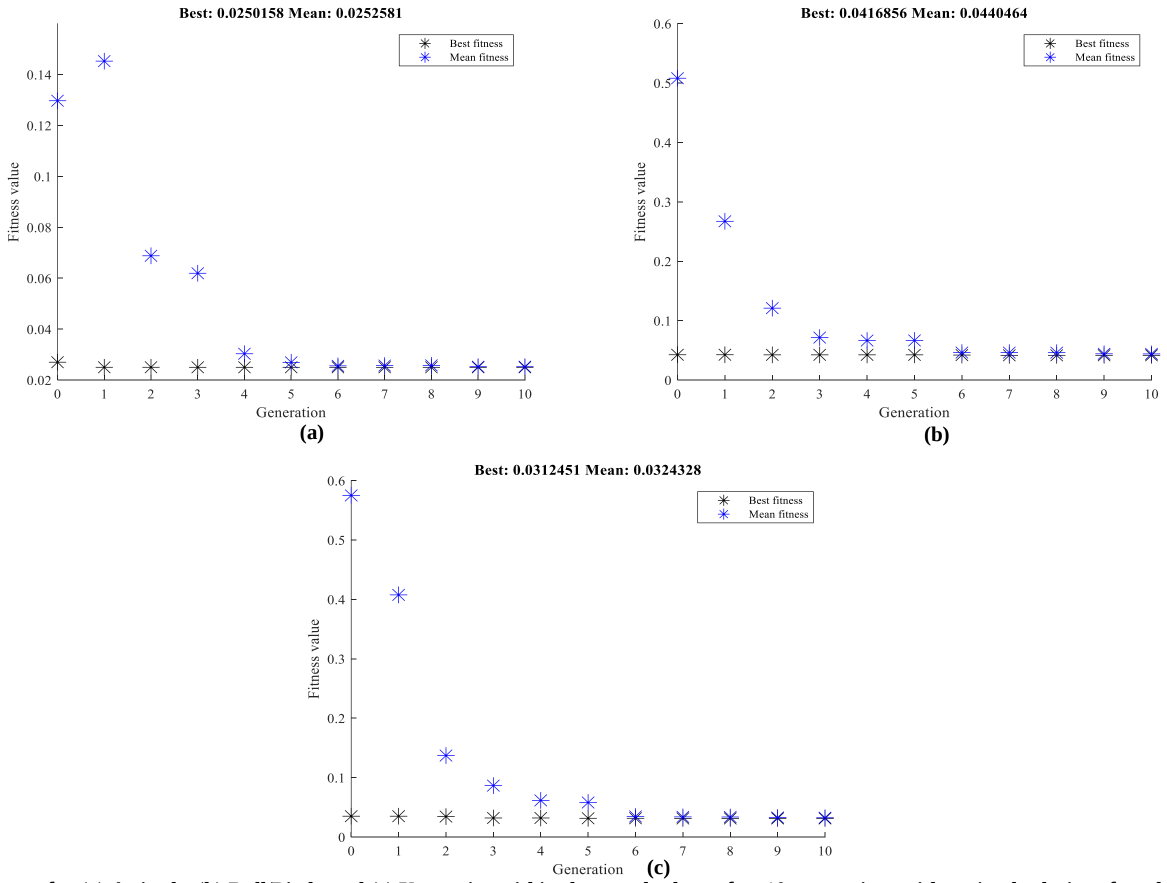

Fig. 10 Search space for (a) Attitude, (b) Roll/Pitch, and (c) Yaw gains within the search plane after 10 generations with optimal solutions found at 10: 0.025372, 0.0305742 and 10: 0.0304205, respectively.

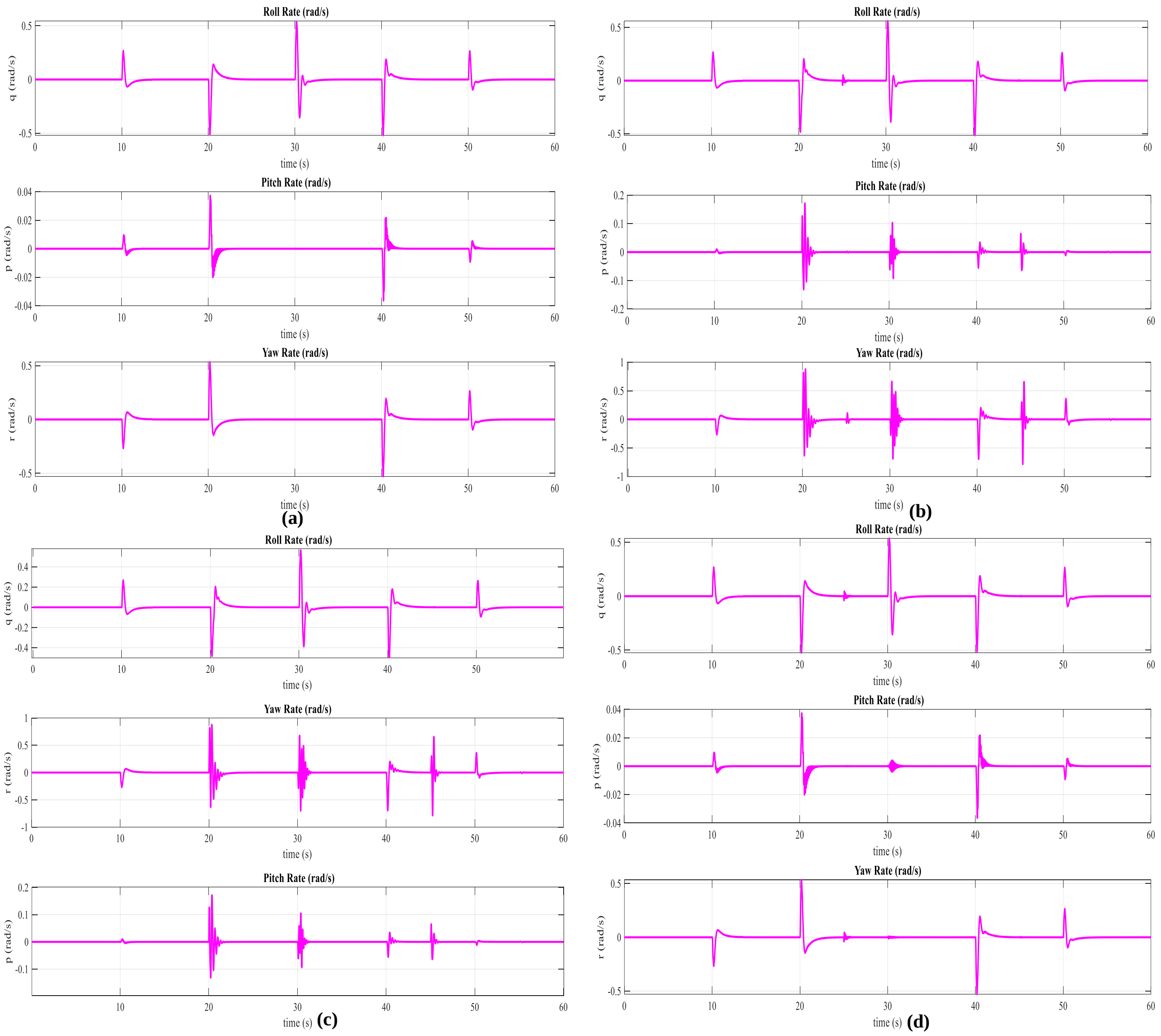

Fig. 11 Motor's Euler angle rates during free-actuator LOE (a) With both actuator LOE and sine wave disturbances, (b) With only actuator LOE, (c) With sine wave disturbances, and (d) At 20th seconds for actuator LOE and 25th for sine wave disturbances.

5. Conclusion

This study investigated a hybrid control scheme for a quadrotor system, utilizing an optimization algorithm based on genetic algorithms to refine solutions for continuously estimated faults. The control scheme employs a PID controller optimized for all state channels integrated with an ADRC to evaluate as well as compensate for system faults and disturbances. The hybrid approach ensures stability and reliability during PID gains optimization and actuator loss compensation. The results showed how well the suggested control technique worked, mainly when it came to managing actuator LOE in a system with only just two actuators functioning correctly-individual tuning for the attitude, roll, pitch, and yaw control channel results in robust performance. The controller was further tested with sine wave disturbances across all control channels to demonstrate the path-tracking capability of the quadrotor UAV. The simulated path-tracking trajectory showed initial deviations that swiftly stabilized with optimized PID gains. Comprehensive tests revealed that actuator faults mainly affected the Yaw state, but overall stabilization of the quadrotor remained effective. The proposed control scheme is versatile and can be applied to different systems with minor modifications.

Acknowledgment

The authors express their gratitude to PAUSTI University for providing financing for this research project and to the Department of Electrical and Electronic Engineering for providing consistent assistance during the project's completion.

References

[1] Bin Xian, and Wei Hao, "Nonlinear Robust Fault-Tolerant Control of the Tilt Trirotor UAV under Rear Servo's Stuck Fault: Theory and Experiments," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2158-2166, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[2] Ademola Abdulkareem et al., "Modeling and Nonlinear Control of a Quadcopter for Stabilization and Trajectory Tracking," *Journal of Engineering*, vol. 2022, pp. 1-19, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[3] J.A. Stankovic, and R. Rajkumar, "Real-Time Operating Systems," *Real-Time Systems*, vol. 28, no. 2-3, pp. 237-253, 2004. [CrossRef] [Google Scholar] [Publisher Link]

[4] Chuanxiang Gao et al., "A UAV-Based Explore-then-Exploit System for Autonomous Indoor Facility Inspection and Scene Reconstruction," *Automation in Construction*, vol. 148, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[5] Iván González-Hernández et al., "Real-Time Improvement of a Trajectory-Tracking Control Based on Super-Twisting Algorithm for a Quadrotor Aircraft," *Drones*, vol. 6, no. 2, pp. 1-20, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[6] Arslan Ahmed Amin, and Khalid Mahmood Hasan, "A Review of Fault Tolerant Control Systems: Advancements and Applications," *Measurement*, vol. 143, pp. 58-68, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[7] Jin Jiang, "Fault-Tolerant Control Systems-An Introductory Overview," *Acta Automatica Sinica*, vol. 31, no. 1, pp. 161-174, 2005. [Google Scholar] [Publisher Link]

[8] Ron J. Patton, "Fault-Tolerant Control: The 1997 Situation," *IFAC Proceedings Volumes*, vol. 30, no. 18, pp. 1029-1051, 1997. [CrossRef] [Google Scholar] [Publisher Link]

[9] Himanshukumar R. Patel, and Vipul A. Shah, "Fuzzy Logic Based Passive Fault Tolerant Control Strategy for A Single-Tank System with System Fault and Process Disturbances," *2018 5th International Conference on Electrical and Electronic Engineering (ICEEE)*, Istanbul, Turkey, pp. 257-262, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[10] Peng Yang et al., "Fault Tolerant PI Control Design for Satellite Attitude Systems with Actuator Fault," *2017 Chinese Automation Congress (CAC)*, Jinan, China, pp. 2026-2030, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[11] Jingqing Han, "From PID to Active Disturbance Rejection Control," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 3, pp. 900-906, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[12] Muhammad Adeel Ahsan et al., "Active Disturbance Rejection Control of a Quadrotor: A Comparative Study," *19th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, pp. 444-450, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[13] Amine Abadi et al., "Robust Tracking Control of Quadrotor Based on Flatness and Active Disturbance Rejection Control," *IET Control Theory & Applications*, vol. 14, no. 8, pp. 1057-1068, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[14] Yusheng Du et al., "Fault Tolerant Control of A Quadrotor Unmanned Aerial Vehicle Based on Active Disturbance Rejection Control and Two-Stage Kalman Filter," *IEEE Access*, vol. 11, pp. 67556-67566, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[15] Lianghao Hua et al., "Sensor Fault Diagnosis and Fault Tolerant Control of Quadrotor UAV Based on Genetic Algorithm," *Journal of Sensors*, vol. 2022, pp. 1-8, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[16] Dapeng Zhang, and Zhiwei Gao, "Fault Tolerant Control Using Reinforcement Learning and Particle Swarm Optimization," *IEEE Access*, vol. 8, pp. 168802-168811, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[17] Seif-El-Islam Hasseni, Latifa Abdou, and Hossam-Eddine Glida, "Parameters Tuning of A Quadrotor PID Controllers by Using Nature-Inspired Algorithms," *Evolutionary Intelligence*, vol. 14, pp. 61-73, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[18] Tzuu-Hseng S. Li, and Ming-Yuan Shieh, "Design of a GA-Based Fuzzy PID Controller for Non-Minimum Phase Systems," *Fuzzy Sets and Systems*, vol. 111, no. 2, pp. 183-197, 2000. [CrossRef] [Google Scholar] [Publisher Link]

[19] Marc Francis Q. Say et al., "A Genetic Algorithm Approach to PID Tuning of A Quadcopter UAV Model," *IEEE/SICE International Symposium on System Integration (SII)*, Iwaki, Fukushima, Japan, pp. 675-678, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[20] Andreas Vikane Hystad, and Joakim Brobakk Lehn, "*Model, Design and Control of a Quadcopter - Implemented on An Arduino Microcontroller, Using Wireless Communication Linked with a Computer Interface, Utilizing Additive Manufacturing Techniques to Enable Simple Replication*," Master of Science in Cybernetics and Robotics Thesis, Norwegian University of Science and Technology, Trondheim, 2015. [Google Scholar] [Publisher Link]

[21] A. Jayachitra, and R. Vinodha, "Genetic Algorithm Based PID Controller Tuning Approach for Continuous Stirred Tank Reactor," *Advances in Artificial Intelligence*, vol. 2014, pp. 1-9, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[22] Peng Lu, and Erik-Jan Van Kampen, "Active Fault-Tolerant Control for Quadrotors Subjected to A Complete Rotor Failure," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, pp. 4698-4703, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[23] Peng Lu, Erik-Jan van Kampen, and Qiping P. Chu, "Nonlinear Quadrotor Control with Online Model Identification," *Advances in Aerospace Guidance, Navigation and Control*, pp. 81-98, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[24] Ata Jahangir Moshayedi, Mahyar Gheibollahi, and Liefa Liao, "The Quadrotor Dynamic Modeling and Study of Meta-Heuristic Algorithms Performance on Optimization of PID Controller Index to Control Angles and Tracking the Route," *IAES International Journal of Robotics and Automation*, vol. 9, no. 4, pp. 256-270, 2020. [CrossRef] [Google Scholar] [Publisher Link]