PROGRESS IN AEROSPACE AND AVIATION TECHNOLOGY
e-ISSN: 2821-2924

PAAT

Vol. 4 No. 2 (2024) 36-50
https://publisher.uthm.edu.my/ojs/index.php/paat

Flight Controller Design for a Helicopter Based Unmanned Aerial Vehicle (UAV) Using Differential Evolution-Based PID Tuning

Illia Mohammed Shukor[1], Syariful Syafiq Shamsudin[1]*, Mohammad Fahmi Pairan[1]

[1] *Research Center for Unmanned Vehicle (RECUV), Faculty of Mechanical and Manufacturing Engineering, University Tun Hussein Onn Malaysia (UTHM), Batu Pahat, 86400, MALAYSIA*

*Corresponding Author: syafiq@uthm.edu.my
DOI: https://doi.org/10.30880/paat.2024.04.02.004

Article Info

Received: 29 January 2024
Accepted: 19 June 2024
Available online: 24 December 2024

Keywords

Helicopter UAV, differential evolution, PID, LQ cost function, multi-objective cost function

Abstract

Helicopters possess the ability to perform a range of maneuvers, including vertical takeoff, hovering, forward or sideways movement, spinning, and turning. Helicopter UAVs are highly advantageous for both military and civilian applications due to their exceptional agility, particularly in hazardous environments. It is critical to evaluate the control system's performance and safety through simulation before using it on actual flights. This helps to reduce risks and minimize the need for extensive flight testing using complex aeronautical systems. The objective of this study is to develop an autonomous flight control system (AFCS) specifically designed for a helicopter-based unmanned aerial vehicle (UAV). This system will control and regulate the helicopter's altitude, attitude, velocity, and heading. The purpose of the study is to determine the optimal gains for a PID controller using the Differential Evolution methodology. Differential evolution is a heuristic method that operates on a population of vectors. To minimize cost functions, it replaces the existing vectors with trial vectors derived from mutant vectors. The optimization process consists of four distinct phases: initialization, mutation, crossover, and selection. The method is a simple and efficient search approach for finding the global optimum. It employs population model functions with real-valued arguments, as well as those defined on completely ordered spaces. The control algorithm underwent testing and simulation using the X-Plane 9 flying simulator and LabVIEW software. A sequence of simulated flight tests validates the performance and effectiveness of the proposed controller. Therefore, it can be inferred that the controller gain obtained by the Differential Evolution (DE) approach in the AFCS design is suitable for hovering and circular trajectory tracking. This is achieved by adjusting the PID gain values based on the Q matrix variations in a multi-objective cost function. The accuracy of position tracking is assessed using the mean absolute error. Both trajectories were measured, yielding a low error and indicating good accuracy.

1. Introduction

The primary goal of implementing an AFCS is to partially automate aircraft flight in order to reduce pilots' workload, particularly during critical flight phases, while ensuring flight safety. AFCS serves two purposes: it enhances the aircraft's inherent flying characteristics, ensuring dynamic stability even in intentionally statically unstable aircraft, and it boosts the aircraft's performance in specific atmospheric conditions [7]. The targeted performance of an unmanned aerial vehicle was modified by adjusting the closed-loop gain settings. PID control, or proportional-integral-derivative feedback control, is a commonly used industrial controller. The core of its success lies in its capacity to efficiently govern many processes and dynamic systems, despite its simple structure and straightforward tuning procedures [7]. To achieve this, a nested structure for the core PID controller is required. PID tuning is the process of selecting the appropriate values for a PID controller's gains to achieve the required performance and meet the design parameters. In essence, the basic objective of implementing a PID controller is to ensure that a measured process value remains at a desired setpoint or target value.

In this literature review, we will focus on several tuning strategies that will be selected to solve the problem statement. The Ziegler-Nichols Frequency Response, Genetic Algorithm, and Differential Evolution (DE) procedures are widely recognized as the most commonly used classical methods for tuning controllers. The differential evolution algorithm (DE) [8] is considered one of the most promising Evolutionary Algorithms (EA) in recent times. Differential Evolution (DE) is an efficient method for global optimization that employs population model functions with real-valued arguments and operates on fully ordered spaces. DE has incorporated simplicity, efficiency, authentic coding, user-friendliness, local search capability, and speed. Differential Evolution (DE) is a versatile method that may be used for various problem sets, including practical ones like data clustering, non-linear function optimization, and image recognition classification [10]. Nevertheless, Differential Evolution (DE) may encounter premature convergence, a situation in which the algorithm reaches a suboptimal solution without fully traversing the whole search space [8]. Cost functions play an important role in selecting the most efficient output decisions. The cost function primarily compares the expected and actual values. The objective of the optimization task is to determine the values of the coefficients K_p, K_i, and K_d that minimize the cost function [5]. The Q matrix in quadratic cost function defines the weights on the states, while the R matrix defines the weights on the control input in the cost function [9]. The cost function used for DE optimization can be used with several objectives. For instance, the Integral Absolute Error (IAE) is a cost function that quantifies cumulative error over time; the Integral Time Square Error (MSE) that is the integral of the product of the squared error over time; or the Integral Time Absolute Error (ITAE) is a cost function that integrates the time multiplied by the absolute value of the error [8].

This study will focus on designing a flight control system for a small helicopter UAV. The system will be developed using the Differential Evolution tuning approach to find the optimal PID gain settings. The flight controller development utilizes a dynamic model that is generated from a linear dynamic system model obtained through system identification work in [14]. This model is utilized to determine suitable parameters for the controller. Ultimately, a series of simulated flight tests are carried out to verify the performance and efficacy of the proposed controller, utilizing the X-Plane flight simulator and incorporating simulated wind disturbances. The proposed controller tuning methods are capable of generating PID controllers that regulate and stabilize the RUAS to the desired performance level and exhibit sufficient robustness in the presence of external disturbances.

2. Kinematics and Dynamics of a Helicopter Based UAV

The kinematics and dynamics model of helicopter-based UAV is described Newton-Euler equations of motion expressed in the Body Reference Frame. The body-fixed reference coordinate is attached to the helicopter's center of gravity (CG) to examine its motion. The x-axis runs along the helicopter's forward direction, the z-axis is downward, and the y-axis is established using the "right-hand" method. The helicopter platform is basically considered a rigid body with body forces, $\boldsymbol{F^B} = \begin{bmatrix} \boldsymbol{F_x^B} & \boldsymbol{F_y^B} & \boldsymbol{F_z^B} \end{bmatrix}^{\boldsymbol{T}}$ and moments, $\boldsymbol{M^B} = \begin{bmatrix} \boldsymbol{M_x^B} & \boldsymbol{M_y^B} & \boldsymbol{M_z^B} \end{bmatrix}^{\boldsymbol{T}}$ are exerted. The twelve dimensional state vectors consist of the helicopter's center of gravity (CG) position vectors in inertial reference frame, $\boldsymbol{p^S} = \begin{bmatrix} \boldsymbol{p_x^S} & \boldsymbol{p_y^S} & \boldsymbol{p_z^S} \end{bmatrix}^{\boldsymbol{T}}$; Euler angles (roll, pitch, and yaw), $\boldsymbol{\Theta^B} = [\boldsymbol{\phi} \quad \boldsymbol{\theta} \quad \boldsymbol{\psi}]^{\boldsymbol{T}}$ in the body reference frame; helicopter linear velocity in the body reference frame, $\boldsymbol{V^B} = [\boldsymbol{u} \quad \boldsymbol{v} \quad \boldsymbol{w}]^{\boldsymbol{T}}$ and helicopter angular velocities in the body reference frame, $\boldsymbol{\omega^B} = [\boldsymbol{p} \quad \boldsymbol{q} \quad \boldsymbol{r}]^{\boldsymbol{T}}$. The overall dynamic of an unmanned rotorcraft system can be divided into kinematics (Eq. (1) - (2)) and system dynamics (Eq. (3) - (4)) as follows:

$$\dot{p}^S = R^{B \to S} V^B \tag{1}$$

$$\dot{\Theta}^S = \Psi(\Theta)\omega^B \tag{2}$$

$$\Psi(\Theta) = \begin{bmatrix} 1 & 0 & tan\,\theta\,cos\,\phi \\ 0 & cos\,\phi & -sin\,\phi \\ 0 & \frac{sin\,\phi}{cos\,\theta} & \frac{cos\,\phi}{cos\,\theta} \end{bmatrix}$$

$$\dot{V}^B = \frac{1}{m}F^B - \omega^B \times V^B \tag{3}$$

$$\dot{\omega}^B = J^{-1}\left(M^B - (\omega^B \times J\omega^B)\right) \tag{4}$$

We can linearize the nonlinear dynamic model in Eq. (1)–(4) near low-speed flight conditions using the steady-state flapping angle approximation [15], [16], which yields the following differential equations:

Attitude Dynamic:

$$\begin{aligned} \dot{\phi} &= p \\ \dot{\theta} &= q \\ \dot{p} &= a_p p + A_p \delta_{lat} + B_p \delta_{lon} \\ \dot{q} &= b_q q + A_q \delta_{lat} + B_q \delta_{lon} \end{aligned} \tag{5}$$

Position and Velocity Dynamics:

$$\begin{aligned} \dot{x} &= u \\ \dot{u} &= X_u u - g\theta \\ \dot{y} &= v \\ \dot{v} &= Y_v v + g\phi \end{aligned} \tag{6}$$

Altitude and Yaw Dynamics:

$$\begin{aligned} \dot{z} &= w \\ \dot{w} &= Z_w w + Z_u \delta_{col} \\ \dot{\psi} &= r \\ \dot{r} &= -c_r r + C_R \delta_{ped} \end{aligned} \tag{7}$$

The control inputs are defined as, $\delta = [\delta_{lon} \quad \delta_{lat} \quad \delta_{col} \quad \delta_{ped}]^T$ where δ_{col} and δ_{ped} are the collective pitch of the main rotor and tail rotor, and δ_{lon} and δ_{lat} are the longitudinal pitch and lateral cyclic pitch, which control the inclination of the main rotor's tip path plane according to their respective directions. Typically, the control input commands are normalized between -1 to +1 for lateral cyclic, longitudinal cyclic, main rotor and tail rotor's collective pitch command. The helicopter UAV's throttle input, on the other hand, is automatically adjusted to keep the rotor speed constant with a value between 0 to +1 range.

Derivatives a_p and b_q are the roll and pitch moment derivatives with respect to flapping angles. Derivatives A_p, B_p, A_q, and B_q are the control derivatives, in which B_p and A_q are used to represent the cross-coupling effect of the cyclic movement on the swashplate. The derivatives X_u, Y_v, and Z_w are known as velocity damping derivatives, which represent the relative fuselage drag forces, g is the acceleration due to gravity force; and Z_u represents the collective pitch control derivative. Note that c_r is the yaw damping derivative used to account for the yaw rate feedback from the onboard heading gyro, and C_R is the yaw control derivative. Table 1 presents the stability derivative values obtained through system identification work in [15].

Table 1 *Identified values for the unknown stability derivatives from system identification experiments [15]*

Dynamic model parameters	Identified value
c_r	3.25
C_r	63.03
b_q	-3.25
B_q	19.5
A_q	6.5

a_p	-3.25
A_p	19.5
B_p	4.55
X_u	-0.178
Y_v	-0.310
Z_w	-2.671
Z_u	-17.81

3. Software in The Loop (SITL) Simulation

Software-in-the-loop (SITL) testing is a process for verifying and testing code in a simulated environment with the goal of efficiently identifying errors and raising the code's quality so that it mimics reality. Iterative testing and design processes mirror those of a real-world system [11]. To fully test the proposed flight controller, a SITL simulation consisted of two primary elements: the LabVIEW control program, which is responsible for executing the control system algorithm to direct the flight control surfaces of the rotorcraft, and the X-Plane flight simulator, which accurately replicates the behavior of the rotorcraft-based UAV, as depicted in Fig. 1. The selection of the X-Plane flight simulator was based on its accuracy in the dynamic response prediction of aircraft models used in the simulator. Furthermore, the Federal Aviation Administration (FAA) has certified X-Plane flight simulators to serve as aids in pilot training, offering a diverse range of aircraft models.

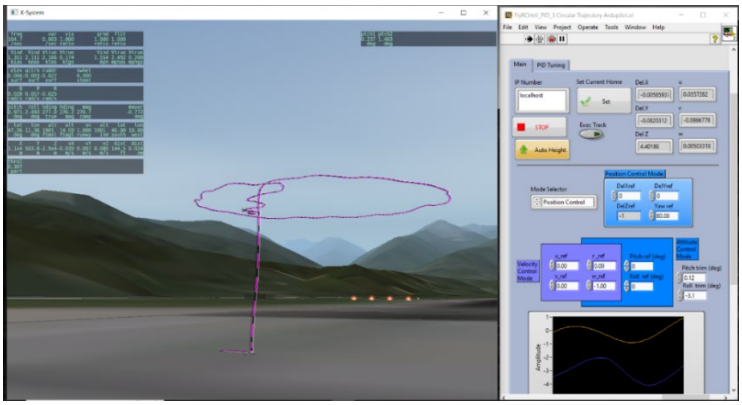


Fig. 1 *X-Plane simulation with TREX 450 helicopter model and LabVIEW program*

X-Plane can be interfaced easily by transmitting the data from numerous flight data packets to the host computer over the Universal Data Packet (UDP) communication protocol. Users can create a LabVIEW program to read the received UDP data and send input commands to the X-Plane software. In this program, the IP address "127.0.0.1," which is the network card's internal address, was used to generate communication between X-Plane and LabVIEW on a single computer [13]. Fig. 2 displays input and output that have been selected to be displayed in the X-Plane Window.

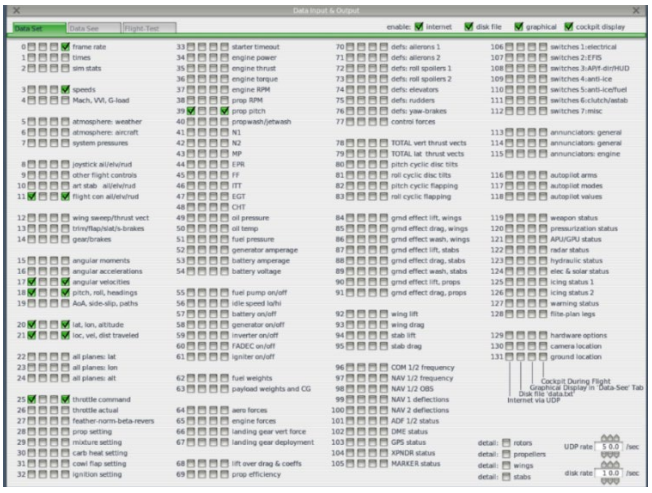


Fig. 2 *Input and output data selection in the X-Plane Data I/O window. The selected flight sensor data is used as feedback to the controller. Controller commands from LabVIEW are also sent back to X-Plane through UDP*

Progress in Aerospace and Aviation Technology Vol. 4 No. 2 (2024) p. 36-50

3.1 UAV Model Description

An electric-powered TREX 450, as shown in Fig. 3, with the features described in Table 2, was employed in this study. A tiny helicopter-based UAV was chosen because of its advantageous applications, including higher agility, mobility, noise reduction, and improved adaptability for forest, urban searches, and indoor flight. On TREX 450, future studies and technological innovations, including swarming, formation flying, and vision-based control, might be tested. The stabilizer bar on the TREX 450 helicopter acts as a damper to reduce the very sensitive aerodynamic forces brought on by the helicopter's incredibly small size and to enable manual control.

Fig. 3 *TREX 450 helicopter model*

Table 2 *Specification of TREX 450*

Specification	TREX 450
Fuselage Length	685 mm
Fuselage Width	162 mm
Fuselage Height	228 mm
Main Rotor Span	680 mm
Tail Rotor Span	150 mm
Blade Number	2
Flight Endurance	12 mins
No-load Weight	600 g
Max Take-Off Weight (MTOW)	800 g
Power Source	Electric engine

4. Flight Controller Structure and Controller Design

The helicopter's complexity often necessitates the reduction and decoupling of its dynamics to streamline the flight control design across various modes and systems. Generally, the primary goal of designing an autopilot is to control the UAV's position ($\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$) and attitude ($\boldsymbol{\phi}$, $\boldsymbol{\theta}$, $\boldsymbol{\psi}$). We can separate the autopilot design into the inner loop and outer loop controllers. To stabilize the UAV's orientation, the outer loop controller controls the translational or velocity dynamics, whereas the inner loop controller controls the attitude dynamics. The control architecture for longitudinal position, forward speed, pitch angle, and pitch rate control is shown in Fig. 9, where the outer loop controller receives the position reference commands $\boldsymbol{x_{ref}}$ before passing the control signal to the forward speed controller. The forward speed controller produced the required error correction and passed the desired reference for the inner-loop controllers (pitch angle and pitch rate controllers). Figure 10 illustrates a similar configuration for controlling lateral position, side speed, roll angle, and roll rate. Fig. 11 provides the control architecture for the altitude, vertical speed, heading hold, and yaw rate controllers. The controllers require the $\boldsymbol{z}$, $\boldsymbol{w}$, $\boldsymbol{\psi}$, and $\boldsymbol{r}$ state feedback from the simulation model.

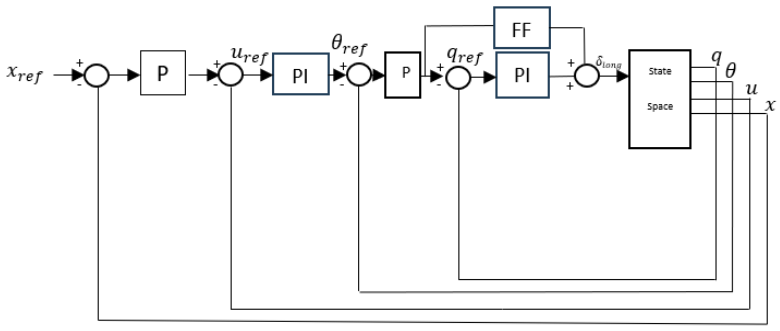


Fig. 9 *Control architecture for longitudinal position, forward speed, pitch angle, and pitch rate control consisting of* $\boldsymbol{x}$, $\boldsymbol{u}$, $\boldsymbol{\theta}$, *and* $\boldsymbol{q}$ *states feedback. The symbols* $\boldsymbol{P}$, $\boldsymbol{PI}$ *and* $\boldsymbol{FF}$ *represent proportional, proportional-integral, and feedforward gains.*

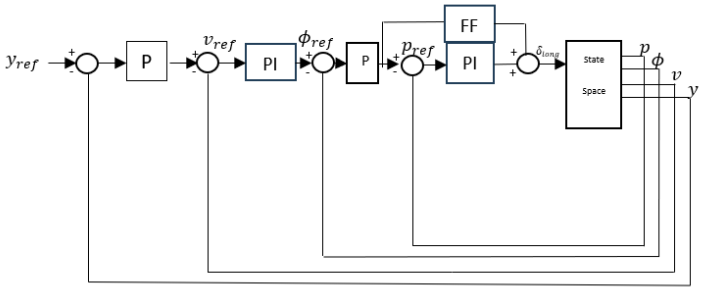


Fig. 10 *Control architecture for lateral position, side speed, roll angle, and roll rate control consisting of* $\boldsymbol{y}$, $\boldsymbol{v}$, $\boldsymbol{\phi}$ *and* $\boldsymbol{p}$ *states feedback. The symbols P, PI and FF represent proportional, proportional-integral, and feedforward gains.*

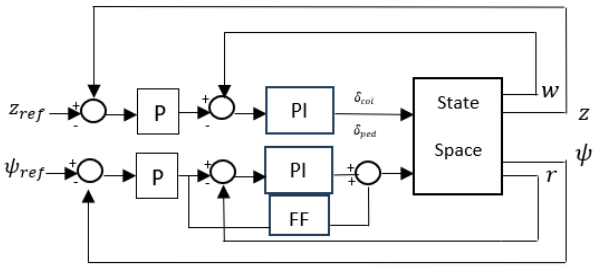


Fig. 11 *Control architecture for the altitude, vertical speed, heading hold and yaw rate control consisting of z, w, ψ, r states feedback. The symbols P, PI and FF represent proportional, proportional-integral, and feedforward gains.*

5. Differential Evolution

Differential evolution is a population-based heuristic method that gradually replaces population vectors with trial vectors made from mutant vectors in order to keep costs as low as possible. Each vector in the population forms mutant vectors by combining a random population vector with a fraction of the difference between two other population vectors. Fig. 12 shows the steps of initialization, mutation, crossover, and selection used to solve optimization problems in DE.

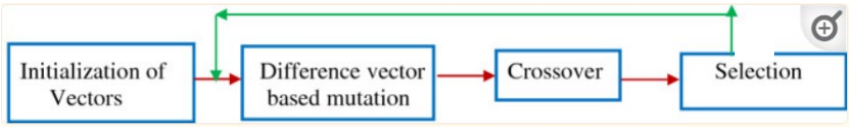


Fig. 12 *Phases of DE*

Trial vectors are created by crossing mutant vectors with population vectors. At the end of each iteration, the trial vectors replace the population vectors whenever they obtain a lower cost function value. We can formulate the unconstrained optimization problem of finding a function's extremum as follows:

$$\begin{gathered} \min \boldsymbol{f}(\boldsymbol{X_1}, \boldsymbol{X_2}, \ldots, \boldsymbol{X_3}) \\ \boldsymbol{s.t} \\ \boldsymbol{X_j^L} \leq \boldsymbol{X_j} \leq \boldsymbol{X_j^U}, \boldsymbol{j} = \boldsymbol{1}, \boldsymbol{2}, \ldots, \boldsymbol{D} \end{gathered} \tag{8}$$

where $\boldsymbol{D}$ indicates the problem's dimension, $\boldsymbol{f}(*)$ denotes the fitness value, $\boldsymbol{X_j^L}$ and $\boldsymbol{X_j^U}$ are, respectively, its minimum and maximum values of $\boldsymbol{X_j}$.

An initial population must be formed in the search space to establish a beginning point. In most engineering challenges, a population size of $NP \approx$ 30-50 is usually sufficient [28]. Without loss of generality, the $\boldsymbol{j^{th}}$ component ($\boldsymbol{j} = \boldsymbol{1}, \boldsymbol{2}, \ldots, \boldsymbol{D}$) of the $\boldsymbol{i^{th}}$ individuals ($\boldsymbol{i} = \boldsymbol{1}, \boldsymbol{2}, \ldots, \boldsymbol{NP}$) in the original population can be expressed as follows:

$$\boldsymbol{X_{i,j}^0} = \boldsymbol{X_{i,j}^L} + \text{rand}(0,1)*\left(\boldsymbol{X_{i,j}^U} - \boldsymbol{X_{i,j}^L}\right) \tag{9}$$

where $\boldsymbol{L}$ and $\boldsymbol{U}$ stand for the lower and upper boundaries of the solution space, respectively, and rand(0,1)returns a uniformly distributed random number between 0 and 1. We need to initialize the DE parameters: scaling or mutation factor $\boldsymbol{F}$, population size, crossover probability $\boldsymbol{CR}$, and maximum generation $\boldsymbol{Gmax}$. Set the iteration

Progress in Aerospace and Aviation Technology Vol. 4 No. 2 (2024) p. 36-50

$\boldsymbol{L} = \boldsymbol{0}$ and then randomly initialized the population $\boldsymbol{X_j^0} = [\boldsymbol{X_1}, \boldsymbol{X_2}, \ldots \boldsymbol{X_{100}}]^{\boldsymbol{T}}$ by using uniform distribution where $\boldsymbol{X} = (\boldsymbol{K_p}, \boldsymbol{K_i}, \boldsymbol{K_d}, \boldsymbol{FF})^{\boldsymbol{T}}$ the parameters of PID.

As depicted in Fig. 13, the population size had been set to 20. Though 20 is typically considered small in DE, it is still considered acceptable. This will lead to a portion of the configuration producing feasible solutions for all executions when searching for the optimized solution. In addition, we set the scale factor, $\boldsymbol{F}$, in LabVIEW to 0.5, as it typically falls within the [0, 1] range. For multi-objective optimization, we set $\boldsymbol{F}$ to the normal distribution random number with expectation 0 and standard deviation 1 as a reference. Moreover, the crossover probability was set to 0.1, given that the range is [0, 1]. Crossover probability ($\boldsymbol{CR}$) has an impact on DE diversity because it controls the number of elements that will change. Less variation in the new population due to smaller $\boldsymbol{CR}$ values will increase exploration within the range of each optimized parameter. However, we must find a compromise value that ensures both local and global search capabilities. The boundary that was set is based on the Mission Planner's range for the helicopter's parameters; for instance, the range for proportional gains for roll, pitch, and yaw angle is between 3 to 12. This range can be obtained under the Full Parameter List in Mission Planner. This boundary ensures that the output will adhere to the limits set for each of these parameters. Lastly, the number of iterations that were set to train the data for all control structures is 5, which is sufficient considering there are only four parameters, namely, $\boldsymbol{K_p}$, $\boldsymbol{K_i}$, $\boldsymbol{K_d}$, and $\boldsymbol{FF}$ to be searched for an optimized output.

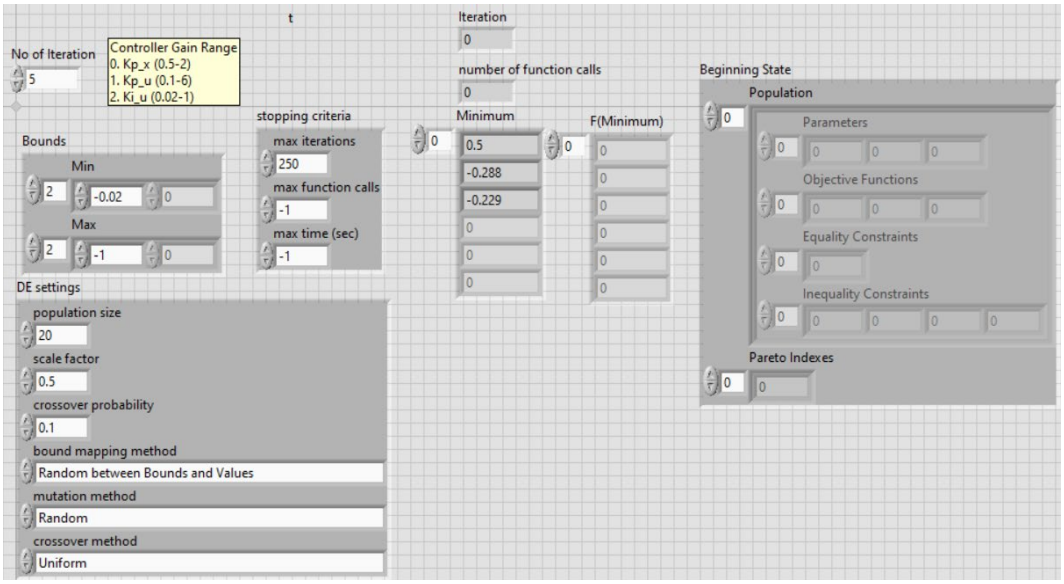


Fig. 13 *Setup parameters of DE in LabVIEW*

The most popular method of carrying out mutations is to choose two individuals at random from the community, scale their vector differences, and then carry out vector synthesis with a third random individual [66]. The mutated individual that was obtained, $\boldsymbol{V_i}$ is as three vectors ($\boldsymbol{X_{r_1}}, \boldsymbol{X_{r_2}}, \boldsymbol{X_{r_3}}$) are chosen at random to have distinct indices $\boldsymbol{i}, \boldsymbol{r_1}, \boldsymbol{r_2}$, and $\boldsymbol{r_3}$. The weighted difference between the two vectors is added to the third vector to produce the donor vector where:

$$\boldsymbol{V_i^{G+1}} = \boldsymbol{X_{r_1}^G} + \boldsymbol{F} * \left(\boldsymbol{X_{r_2}^G} - \boldsymbol{X_{r_3}^G}\right) \tag{9}$$

$\boldsymbol{F}$ denotes the scaling factor and controls the amplification of the differential vector. Choosing an acceptable value for F is critical for achieving the correct balancing of the algorithm's exploration and exploitation searches and avoiding undesirable downsides such as premature convergence or poor convergence pace. A larger $\boldsymbol{F}$ can cause a larger "perturbation," which is beneficial to maintaining population variety but reduces the algorithm's search efficiency as it may allow the algorithm to overshoot good optima. A smaller $\boldsymbol{F}$ contributes to faster convergence, but the loss of population variety is faster, and it is possible to fall into local optimal and premature convergence. Therefore, the value must be small enough to encourage local exploration while simultaneously being large enough to preserve diversity.

The crossover operation's goal is to produce the trial vector. According to reports, exponential crossover performs better on specific types of optimization problems, such as those having links between nearby choice variables [8]. Additionally, the possibility of crossover CR influences DE diversity since it determines the number of elements that may change. Larger CR values will introduce more variation in the new population, thereby

enhancing exploration. However, to ensure both local and global search capabilities, a compromise value must be determined.

$$U_{i,j}^{G+1} = \begin{cases} V_{i,j}^{G+1} & , rand < CR \text{ or } j = j_{rand} \\ X_{i,j}^{G} & , \quad otherwise \end{cases} \tag{10}$$

Once the next generation's population is produced, the iterative processes of mutation, crossover, and selection are repeated indefinitely until the termination requirements are met. The selection process for DE is mathematically described as follows:

$$X_i^{G+1} = \begin{cases} U_{i,j}^{G+1}, & f(U_{i,j}^{G+1}) < f(X_i^G) \\ X_{i,j}^{G} & , \quad \text{otherwise} \end{cases} \tag{11}$$

If the most recent trial vector, yields a higher objective function value, the current target vector X will be substituted by it in the following iteration. DE selection can be carried out in both synchronous and asynchronous ways. During the synchronous mode, the DE population can be updated simultaneously, but the asynchronous mode can be utilized to update the DE population individually [1].

5.1 Cost Function

Generally, cost functions play a vital role in determining optimal output choices. It essentially compares expected and actual values. The purpose of the optimization task is to identify coefficients $\boldsymbol{K_p}$, $\boldsymbol{K_i}$, and $\boldsymbol{K_d}$ that minimize the cost function, while considering the dynamics of the system under control. A simple cost function, known as the linear quadratic (LQ) cost function, is used in optimization problems:

$$J = \int eQ\ e^T + uR\ u^T\ dt \tag{12}$$

where $\boldsymbol{e}$ represents the error, $\boldsymbol{u}$ signifies the control input, and $\boldsymbol{Q}$ and $\boldsymbol{R}$ denote the state-cost weighting and input-cost weighting matrices. The $\boldsymbol{Q}$ matrix defines the weights on the states, whereas the $\boldsymbol{R}$ matrix defines the weights on the control input in the cost function. The choice of $\boldsymbol{Q}$ affects the trade-off between minimizing the state values and minimizing the control effort. Larger values in $\boldsymbol{Q}$ imply a higher cost for deviations in the corresponding state variables, leading to a stronger emphasis on minimizing state errors. However, this selection for $\boldsymbol{Q}$ and $\boldsymbol{R}$ primarily depends on trial and error and the designer's experience. Therefore, in practical applications, the specific characteristics and requirements of the system under control dictate the choice of $\boldsymbol{Q}$. It often involves tuning through experimentation or based on the system's dynamics and performance objectives. We can also use cost functions for multiple objectives, such as IE, a cost function that integrates error, LQ, a linear quadratic cost function, and ITAE, a cost function that integrates time multiplied by the absolute value of error [8]. In this project, we employed the LQ cost function to reduce the tracking error and control effort, and we incorporated a penalty into the cost function to reduce the rise time:

$$J = \int (eQ\ e^T + uR\ u^T)dt + WT_r \tag{13}$$

where $\boldsymbol{W}$ is the weighting constant and $\boldsymbol{T_r}$ is the rise time.

The block diagram for the cost function is depicted in Eq. 13, which eventually will be used to evaluate the performance of the control structure as shown in Fig. 14. The LQ cost function is implemented outside the control and simulation blocks at the top right of the view. In this control structure, the $\boldsymbol{Q}$ and $\boldsymbol{R}$ scalars were set to 0.1 and 1, respectively. A more detailed result from the $\boldsymbol{Q}$ selection will be explained in Section 6. To train the data in DE based on this control structure, it is necessary to call the control structure's LabVIEW file by using a file browser, as shown in Fig. 15, thereby allowing the DE's LabVIEW file to read through and minimize the cost function to get that optimized output.

Progress in Aerospace and Aviation Technology Vol. 4 No. 2 (2024) p. 36-50

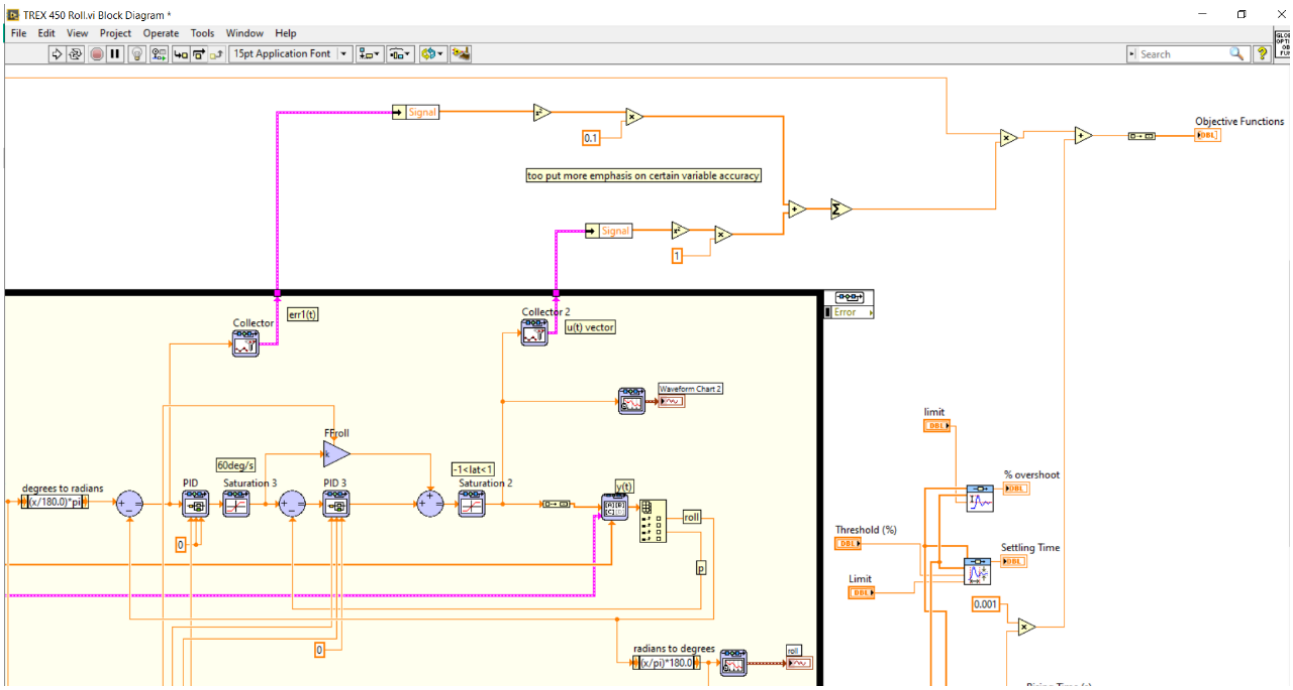


Fig. 14 *Block diagram to portray the cost function connected to control structure*

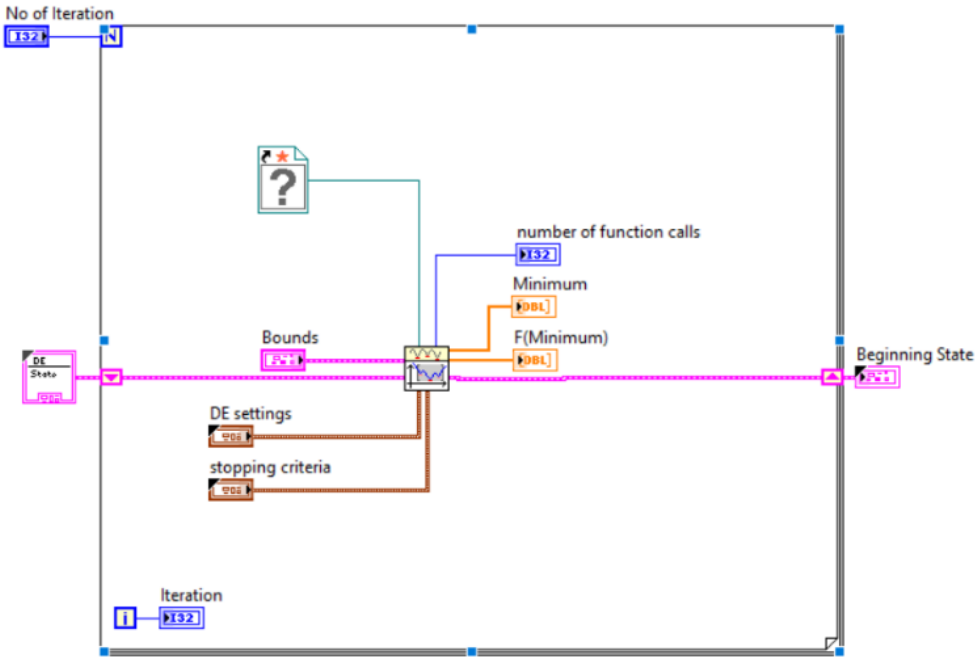


Fig. 15 *Block diagram for DE*

6. Results and Discussion

With the aid of Microsoft Excel and the Google Spreadsheet, the PID gain values were produced and reported below, which show the values of the proportional gain, K_p, integral gain, K_i, derivative gain, K_d and feedforward, FF in Tables 3 and 4. PID gain results were produced using the DE with LQR and a multi-objective cost function implemented, and the resulting control response for roll angle is shown in Fig. 16. Following the acquisition of PID gains, two common trajectory tracking simulations, specifically hovering and circular trajectory were executed in the SITL simulation environment, LabVIEW.

Table 3 *Result of PID gains in DE using initial scalars.*

Control Structure	State Space, A	State Space, B	Desired PID gains	Weighting matrix	PI-controller
Roll	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -3.25 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ -19.5 \end{bmatrix}$	$K_{p_\phi} = 3$ $FF_p = 0.07$ $K_{p_p} = 0.01$ $K_{i_p} = 0.01$	Q = 1 R = 1	$K_{p_\phi} = 3$ $FF_p = 0.141$ $K_{p_p} = 0.159$ $K_{i_p} = 0.01$
Roll Pitch	$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -3.25 & 0 \\ 0 & 0 & 0 & -3.25 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 19.5 & 4.55 \\ 6.5 & 19.5 \end{bmatrix}$	$K_{p_\theta} = 3$ $FF_q = 0.07$ $K_{p_q} = 0.01$ $K_{i_q} = 0.01$	Q = 1 R = 1	$K_{p_\theta} = 3$ $FF_q = 0.141$ $K_{p_q} = 0.159$ $K_{i_q} = 0.01$
Speed Position [x,u]	$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -0.178 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -0.31 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ -9.81 & 0 \\ 0 & 0 \\ 0 & 9.81 \end{bmatrix}$	$K_{p_x} = 0.5$ $K_{p_u} = -0.288$ $K_{i_u} = -0.229$	Q = 1 R = 1	$K_{p_x} = 0.621$ $K_{p_u} = -0.384$ $K_{i_u} = -0.984$
Speed Position [y,v]			$K_{p_y} = 0.5$ $K_{p_v} = 0.274$ $K_{i_v} = 0.229$	Q = 1 R = 1	$K_{p_y} = 0.803$ $K_{p_v} = 0.699$ $K_{i_v} = 0.541$
Yaw	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -3.25 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ -63.3 \end{bmatrix}$	$K_{p_\psi} = 3$ $FF_r = 0.2$ $K_{p_r} = 0.1$ $K_{i_r} = 0.01$	Q = 1 R = 1	$K_{p_\psi} = 3.13$ $FF_r = 0.261$ $K_{p_r} = 0.104$ $K_{i_r} = 0.01$
Yaw Altitude	$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & -2.671 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -3.25 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ -17.81 & 0 \\ 0 & 0 \\ 0 & 63.3 \end{bmatrix}$	$K_{p_z} = 2$ $K_{p_w} = -1$ $K_{i_w} = -0.505$	Q = 1 R = 1	$K_{p_z} = 2.51$ $K_{p_w} = -7.23$ $K_{i_w} = -0.81$

Table 4 *Result of PID gains in DE when Q is 0.1*

Control Structure	Desired PID gains	Weighting matrix	PI- controller
Roll	$K_{p_\phi} = 3$ $FF_p = 0.07$ $K_{p_p} = 0.01$ $K_{i_p} = 0.01$	Q = 0.1 R = 1	$K_{p_\phi} = 3$ $FF_p = 0.02$ $K_{p_p} = 0.08$ $K_{i_p} = 0.01$
Roll Pitch	$K_{p_\theta} = 3$ $FF_q = 0.07$ $K_{p_q} = 0.01$ $K_{i_q} = 0.01$	Q = 0.1 R = 1	$K_{p_\theta} = 3$ $FF_q = 0.02$ $K_{p_q} = 0.08$ $K_{i_q} = 0.01$
Speed Position [x,u]	$K_{p_x} = 0.5$ $K_{p_u} = -0.288$ $K_{i_u} = -0.229$	Q = 0.1 R = 1	$K_{p_x} = 0.5$ $K_{p_u} = -0.291$ $K_{i_u} = -0.259$
Speed Position [y,v]	$K_{p_y} = 0.5$ $K_{p_v} = 0.274$ $K_{i_v} = 0.229$	Q = 0.1 R = 1	$K_{p_y} = 5$ $K_{p_v} = 0.314$ $K_{i_v} = 0.277$
Yaw	$K_{p_\psi} = 3$ $FF_r = 0.2$ $K_{p_r} = 0.1$ $K_{i_r} = 0.01$	Q = 0.1 R = 1	$K_{p_\psi} = 3$ $FF_r = 0.123$ $K_{p_r} = 0.1$ $K_{i_r} = 0.01$
Yaw Altitude	$K_{p_z} = 2$ $K_{p_w} = -1$ $K_{i_w} = -0.505$	Q = 0.1 R = 1	$K_{p_z} = 1.1$ $K_{p_w} = -1.15$ $K_{i_w} = -0.65$

When the value of $\boldsymbol{Q} = \mathbf{0.1}$, the majority of the controller values are smaller compared to when $\boldsymbol{Q} = \mathbf{1}$. The utilization of the acquired gain values resulted in improved stability for the helicopter UAV during flight simulator testing, as it effectively minimized oscillation. Hence, the selection of the scalar $\boldsymbol{Q}$'s weighting is crucial as it has the potential to impact how close the outcome to the desired value. The roll response performance was determined by implementing the PID gains from Table 3 and Table 4 in LabVIEW, as shown in Fig. 16. Hence, the result of the performance is documented in Table 5. Both scenarios, where $\boldsymbol{Q}$ is set at 0.1 and 1, are able to reach a stable state within a time frame of 10 seconds. Furthermore, the time it takes for the system to reach a value of 1 when $\boldsymbol{Q} = \mathbf{1}$ is shorter, specifically 2.31 seconds, compared to when $\boldsymbol{Q} = \mathbf{0.1}$, which takes 3.09 seconds. In Figure 18, a small overshoot is observed when PID gain settings with $\boldsymbol{Q} = \mathbf{1}$ are used, compared to the response when $\boldsymbol{Q} = \mathbf{0.1}$. In this case, the overshoot is greater. Although the overshoot is greater when $\boldsymbol{Q} = \mathbf{0.1}$ compared to $\boldsymbol{Q} = \mathbf{1}$, the system exhibits no oscillation, which is preferable for achieving substantial stability in the flight of the helicopter UAV. The oscillation observed at $\boldsymbol{Q} = \mathbf{1}$ is likely caused by the significant proportional gain acquired, resulting in system instability. The choice of $\boldsymbol{Q}$, the state cost matrix, directly impacts the performance of the helicopter's roll by influencing the PID gain values. This option helps in achieving gain values that are considerably closer to the intended values.

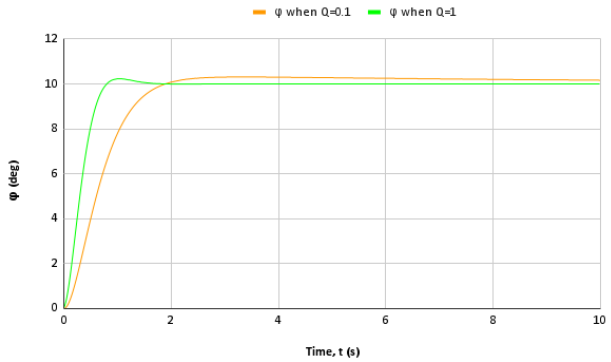


Fig. 16 *Roll response against time*

Table 5 *Comparison in performance for roll angle when $\boldsymbol{Q}$ varies*

Performance Characteristic	φ when Q = 1	φ when Q = 0.1
Reference angle, (deg)	10	10
Settling Time, Ts (s)	10	10
Rising Time, Tr (s)	0.5	1.12
Overshoot, OS (%)	2.31	3.09

Executing the PID gains in Table 6 in LabVIEW yielded the roll response performance. Consequently, Table 7 records the performance outcome. When the rising time is applied to the control structure for roll, both conditions can get a similar settling time of 10 seconds. In addition, the application of the multi-objective cost function

Progress in Aerospace and Aviation Technology Vol. 4 No. 2 (2024) p. 36-50

reduces the rising time to 0.82 seconds compared to its absence. Fig. 17 shows a slight overshoot when applying the rising time, compared to the response without it. Despite the overshoot increasing to 6.6% with the rising time, it remains acceptable due to the control structure's 10% overshoot limit. Thus, multi-objective optimization with additional objectives does affect the performance of the helicopter's roll in terms of its obtained PID gain values, where it assists in achieving a closer value to the desired gain values.

Table 6 *Result of PID gains in DE when* $\boldsymbol{T_r}$ *is applied*

Control Structure	Desired PID gains	Weighting of Tr	PI- controller	
			Without Tr	With Tr
Roll	$K_{p_\phi} = 3$ $FF_p = 0.07$ $K_{p_p} = 0.01$ $K_{i_p} = 0.01$	W = 0.001	$K_{p_\phi} = 3$ $FF_p = 0.02$ $K_{p_p} = 0.08$ $K_{i_p} = 0.01$	$K_{p_\phi} = 3$ $FF_p = 0.09$ $K_{p_p} = 0.016$ $K_{i_p} = 0.01$

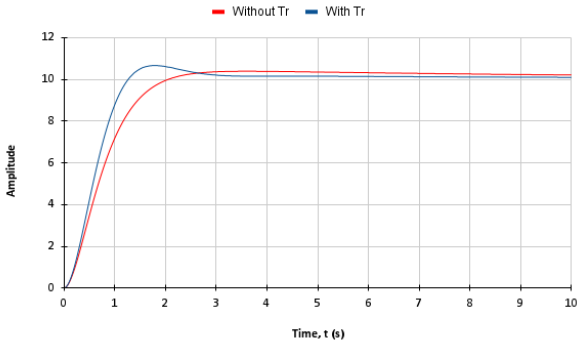


Fig. 17 *Roll Response without and with rising time*

Table 7 *Difference in performance for roll response*

Performance Characteristic	Without Tr	With Tr
Settling Time, Ts (s)	10	10
Rising Time, Tr (s)	1.24	0.82
Overshoot, OS (%)	3.86	6.6

6.1 Results for Hover Trajectory

The simulation result displayed in Fig. 18 indicates that the hovering trajectory tracking is rather straight and smooth when tested in the flight simulator. Even though there is a lot of fluctuation in Fig. 19(b) and Fig. 19(c) for each of the controller servo inputs, the helicopter's movement is still regarded as stable. Cyclic longitudinal control allows for horizontal movement of the helicopter during hover flight, while cyclic lateral control allows for forward and backward movement. The longitudinal cyclic input graph displayed in Fig. 19(a) indicates that the elevator servo input range for the pole placement approach is between -0.02 and 0.005. On the other hand, the aileron servo input for the DE is in the range of -0.01 to 0.0025, according to the lateral cyclic input graph displayed in Fig. 19(b). Also, in Fig. 20(a), the helicopter managed to achieve a height of 10 m as set for hovering. As for the attitude input response graph in Fig. 20, although at the beginning, the response of roll and pitch angle oscillates upon reaching 5 seconds, it gets steady once it has reached 10-meter height during hover. Since the reference Z-position is 10 meters and the actual Z-position is obtained in Fig. 20(b), error can be determined by using the formula for mean absolute error (MAE), where MAE is the MAE is the average absolute error between actual and predicted values, as shown in Equation 14 [2]. This formula can be used to measure the accuracy of hover tracking.

$$\bar{e} = \frac{\sum|(Y_d - Y)|}{N} \tag{14}$$

where $\bar{e}$ is the mean absolute error, Y_d is the desired value, Y is the actual value, and N is the number of data points. With the aid of Microsoft Excel, the error that has been obtained from the formula is 0.01. It is clear that the error is very low; hence, it suggests good accuracy and precision in the hovering tracking. In many applications, a low mean absolute error is desirable, as it indicates that, on average, the actual values are not too far from the expected values [2].

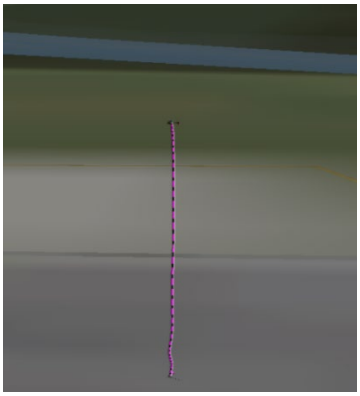


Fig. 18 *Hovering Simulation in X-Plane*

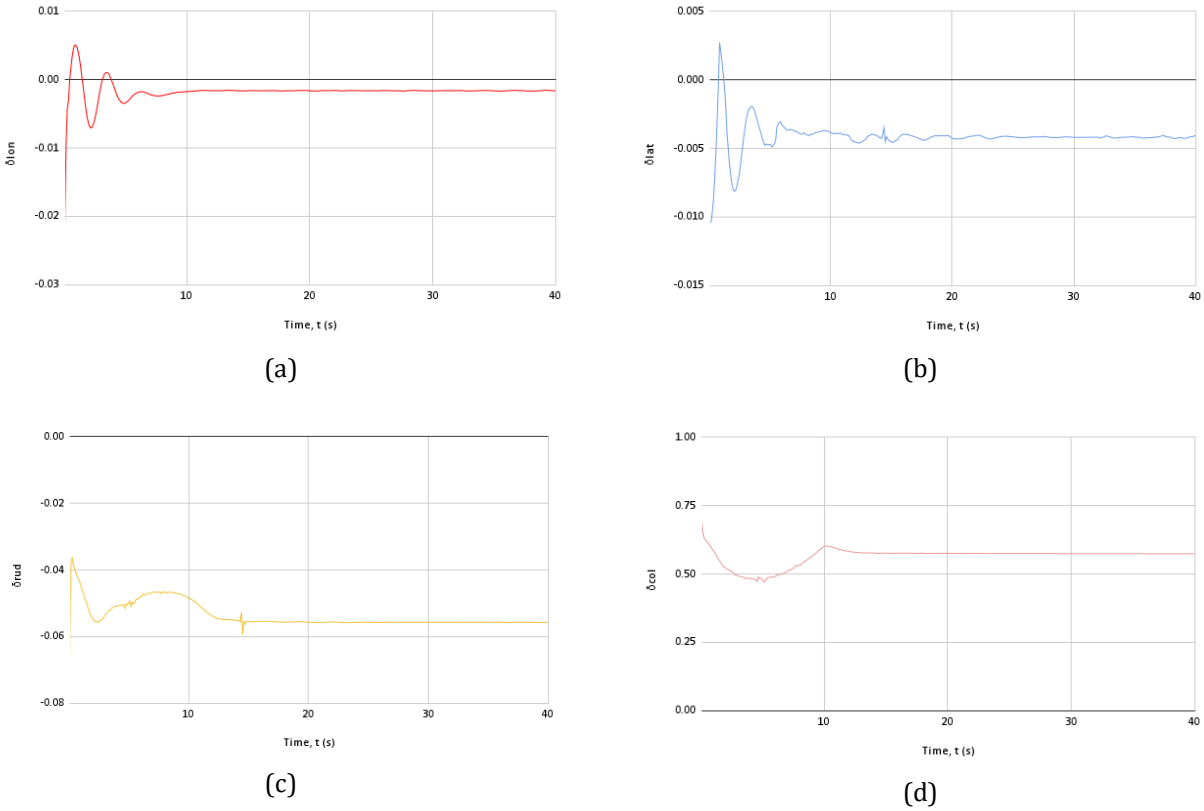


Fig. 19 *PID controller tuned by DE for hovering tracking produces the controller input response where (a) Longitudinal cyclic Collective input (b) Lateral cyclic Rudder input (c) Rudder input (d) Collective input*

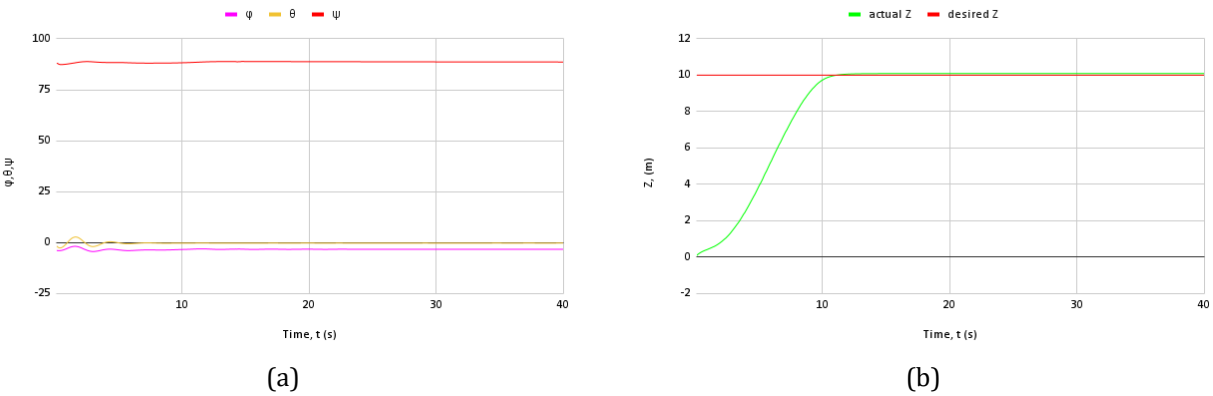


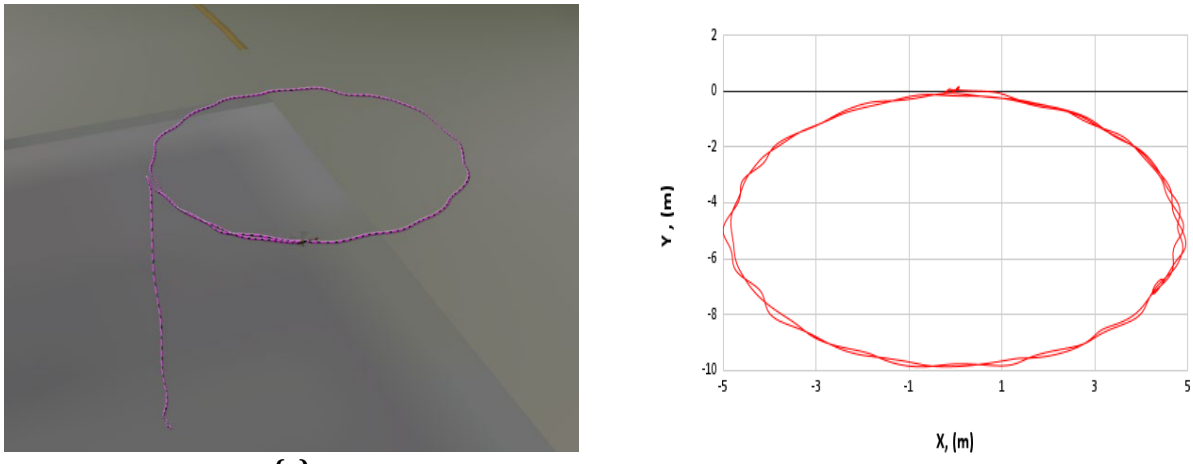
Fig. 20 *Position input response graph generated by the PID controller that DE adjusted for hovering tracking where (a) Attitude input response graph; (b) Z input response graph to get error*

Progress in Aerospace and Aviation Technology Vol. 4 No. 2 (2024) p. 36-50

6.2 Results for Circular Trajectory

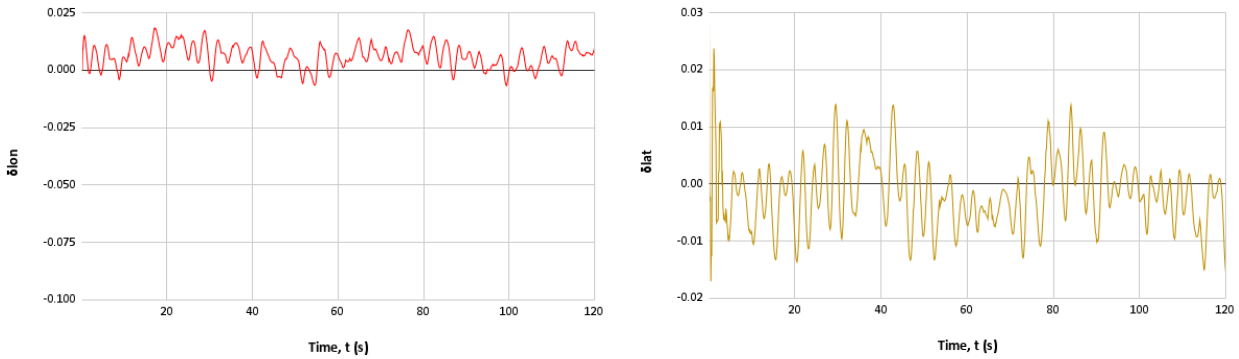
Cyclic longitudinal control is used to create side movement; cyclic lateral control is used to adjust the rotorcraft's forward speed and controlled turn; main rotor collective control is used to adjust power using the rotor blade pitch setting; and tail rotor collective control is used to adjust the rotorcraft's sideslip angle. Based on the simulation result displayed in Fig. 21, we can see that the helicopter UAV's trajectory tracking is not so good, but this is also considering the presence of wind disturbance. It initially goes off course during its first circular tracking attempt, but on its second tracking attempt, it returns to normal. But much like with the previous controller, the helicopter UAV's circular trajectory tracking using the DE approach is also rather good; however, during its second circle tracking, it returns to normal after a short off-track moment.

The longitudinal cyclic input graph in Fig. 22(a) reveals that the elevator servo input range for the DE approach falls between -0.08 and 0.1. However, the Aileron servo input for the DE approach is between -0.02 and 0.02, according to the lateral cyclic input graph displayed in Fig. 22(b). As for Fig. 23, due to the presence of a wind disturbance, the yaw angle is fluctuating around 80 degrees, but it is still within the reference range. Since the reference yaw angle is 80° and the actual yaw angle is obtained in Fig. 24, error can be determined by using the formula for mean absolute error (MAE), where MAE is the MAE is the average absolute error between actual and predicted values [2], as shown in Equation 26. With the aid of Microsoft Excel, the error that has been obtained from the formula is 0.001. It is clear that the error is very low; hence, it suggests good accuracy and precision in the circular tracking, considering there are wind disturbances as well.

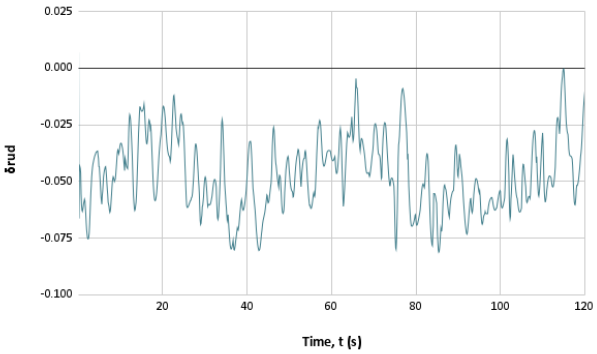


(a) **(b)**

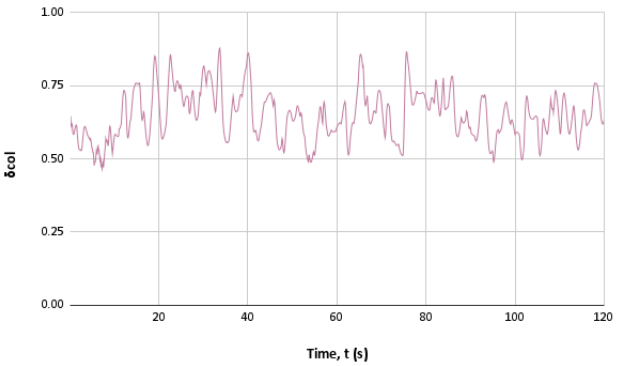
Fig. 21 *Circular Simulation with wind disturbance (a) Circular trajectory simulation X-Plane (b) x-y position response*



(a) **(b)**



(c)



(d)

Fig. 22: *PID controller tuned by DE for circular tracking produces the controller input response where (a) Longitudinal cyclic Collective input (b) Lateral cyclic (c) Rudder input (d) Collective input*

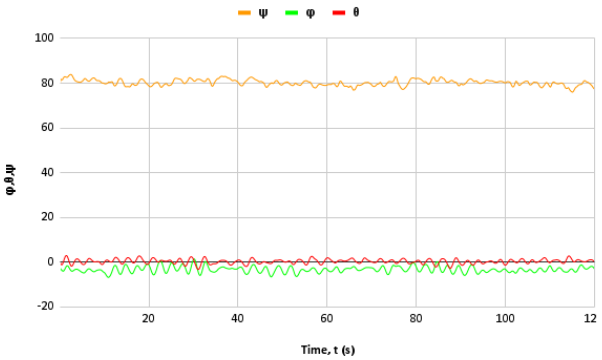


Fig. 23 *The graph for ϕ, θ and ψ responses against time*

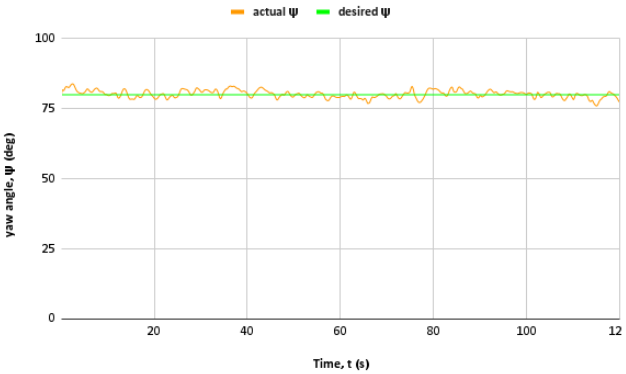


Fig. 24 *Yaw angle response graph*

7. Conclusion

This study investigates the reliability of utilizing the Differential Evolution approach, in conjunction with the LQ cost function and multi-objective optimization, to adjust the PID gain values of a helicopter UAV. The objective is to evaluate the unmanned aerial vehicle's efficiency in X-Plane 9 without the possibility of accidents during real-world trials using the SITL simulation. This work examines the theoretical basis of an Automatic Flight Control System (AFCS) that utilizes the Differential Evolution (DE) tuning mechanism. The study successfully fulfills its goals by utilizing Differential Evolution (DE) to develop an autonomous flight control system (AFCS) for a helicopter-based UAV. The AFCS incorporates essential functionalities such as altitude, velocity, attitude, and heading control, along with waypoint trajectory tracking. Furthermore, the study investigates how the cost function in DE optimization affects the flight controller's performance. It utilizes a multi-objective cost function in the roll control structure to minimize the time it takes for the aircraft to reach its desired position. The results of our investigation indicate that the selection of the Q matrix in the LQ formula and the inclusion of a multi-objective cost function in the DE approach have a substantial impact on the controller's ability to accurately follow circular trajectories. Using a value of $\boldsymbol{Q} = \mathbf{1}$ for the roll angle controller in X-Plane is inappropriate and will result in significant oscillations in roll response. On the other hand, using $\boldsymbol{Q} = \mathbf{0.1}$ resulted in less oscillation and increased stability. It is possible to increase the Differential Evolution Tuning method's capability by experimenting with various PID controller multi-objective cost functions.

Acknowledgement

This research was supported by Universiti Tun Hussein Onn Malaysia (UTHM) through Tier 1 (Vot. Q112) research grant.

Conflict of Interest

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

The authors confirm contribution to the paper as follows: ***study conception and design:*** *Syariful Syafiq Shamsudin, Mohammad Fahmi Pairan;* ***data collection:*** *Illia Mohammed Shukor;* ***analysis and interpretation of results:*** *Illia Mohammed Shukor;* ***draft manuscript preparation:*** *Illia Mohammed Shukor, Syariful Syafiq Shamsudin. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] Ahmad, F., Mat Isa, N., Lim, W. H., & Ang, K. M. (2022). Differential evolution: A recent review based on state-of-the-art works, 61(5), 3831-3872. https://doi.org/10.1016/j.aej.2021.09.013A

[2] A. Jayachitra, R. Vinodha, "Genetic Algorithm Based PID Controller Tuning Approach for Continuous Stirred Reactor", Advances in Artificial Intelligence, vol. 2014, Article ID 791230, 8 pages, 2014. https://doi.org/10.1155/2014/791230

[3] Allwright, S. (2022). What is a good MAE score? (simply explained). Stephen Allwright. https://stephenallwright.com/good-mae-score/

[4] Alvarenga, J., Vitzilaios, N. I., Valavanis, K. P., & Rutherford, M. J. Survey of unmanned helicopter model-based navigation and control techniques. Journal of Intelligent & Robotic Systems, (2014). 80(1), 87-138. https://doi.org/10.1007/s10846-014-0143-5

[5] Bansal H.O., Sharma R., Shreeraman P. PID controller tuning techniques: a review. J Control Eng Technol. 2012;2(4):168–176

[6] Bujok, P. (2021). The real-life application of differential evolution with a distance-based mutation-selection. Mathematics, 9(16), 1909. https://doi.org/10.3390/math9161909

[7] M Islam, M Okasha, and M M Idres, "Dynamics and control of quadcopter using linear model predictive control approach," IOP Conf. Ser.: Mater. Sci. Eng. 270 012007, 2017, https://doi.org/10.1088/1757-899X/270/1/012007

[8] NI. (2023, July 5). Defining a Cost Function (Control Design and Simulation Module). LabVIEW Control Design and Simulation Module. https://www.ni.com/docs/en-US/bundle/labview-control-design-and-simulation-module/page/lvsimconcepts/sim_c_costfunc.html#:~:text=IAE%E2%80%94A%20cost%20function%20that,time%20multiplied%20by%20the%20error

[9] Ogidi-Ekoko, O. N., Liang, W., Xue, H., & Tansu, N. (2021). Machine learning inspired the design of complex-shaped Gan Subwavelength grating reflectors. IEEE Photonics Journal, 13(1), 1-13. https://doi.org/10.1109/jphot.2020.3048182 3.2.1

[10] Piotrowski, A., Napiorkowski, J. J., & Agnieszka Piotrowska. (2023). Particle Swarm Optimization or Differential Evolution—A comparison. Engineering Applications of Artificial Intelligence, 121, 106008–106008. https://doi.org/10.1016/j.engappai.2023.106008

[11] Sir Elkhatem, A., & Naci Engin, S. (2022). Robust LQR and LQR-PI control strategies based on adaptive weighting matrix selection for a UAV position and attitude tracking control. Alexandria Engineering Journal, 61(8), 6275–6292. https://doi.org/10.1016/j.aej.2021.11.057

[12] Study, F. (2019). Forces acting on the helicopter. Flying Training. https://www.flight-study.com/2019/12/forces-acting-on-aircraft.html

[13] X-plane UDP data input/output via LabVIEW. (2013, September 10). Anas's Blog. https://anasbiniftikhar.wordpress.com/2013/09/10/xplane-udp-data-inputoutput-via-labview/

[14] Y. Ben, "Development of a miniature low-cost UAV helicopter autopilot platform and formation flight control of UAV teams," Department of Electrical and Computer Engineering, National University of Singapore, 2010.

[15] Cunjia Liu, "Advanced control for miniature helicopters: modelling, design and flight test," Doctoral Thesis, Loughborough University, 2011. Accessed: Jan. 01, 2023. [Online]. Available: https://repository.lboro.ac.uk/account/articles/9219725

[16] C. Liu and W. Chen, "Dynamics Modelling and System Identification of Small Unmanned Helicopters," in *Advanced UAV Aerodynamics, Flight Stability and Control*, Wiley, 2017, pp. 255–282. https://doi.org/10.1002/9781118928691.ch7