EAI Endorsed Transactions
on Creative Technologies

Research Article EAI.EU

Quadcopter Vehicle Proportional-Integral-Derivative Controller (PIDC) Tuning by Particle-Swarm-Optimization (PSO)

Abdullah Al Gizi[1,*]

[1]Thi-Qar Technical Collage, Dept. Of Electromechanical Engineering, Southern Technical University, Thi-qar, Iraq

Abstract

Inspection of the faults and damages in high voltage transmission lines is not only unsafe and costly but also a very time-consuming process. In addition, it demands highly skilled manpower for the operation at tens of meters above the ground on cables carrying thousands of Volts. Meanwhile, the Quadcopter Aerial Vehicle (QAV) system became a popular alternative for aerial photography applications due to its low cost, ease of usage, and fast response wherein the proportional-integral-derivative controller (PIDC) is used. The PID controller compares the expected location of the drone with the actual measured position thereby can accurately detect the faults in the transmission lines. Based on these factors, in this study, the flight simulation is done according to the desired flight path, and drone imagery is used for (insulator, power line, and porcelain) flaw identification. Furthermore, three movements (roll, pitch, and yaw) of the quadcopter were controlled by different PIDC that were optimized using the particle swarm optimization (PSO). The MATLAB/Simulink application was used to develop the system and simulate the results (R2021b). This clearly suggested that the quadcopter continues to fly on a trajectory with minimum inaccuracy, wherein the PIDC operated as a closed-loop adaptive controller. Finally, the PSO-PIDC outperformed the PIDC due to the critical nature of manual gain modification for quadcopter flight stability.

Keywords: Proportional-Integral-Derivative Controller (PIDC); Quadcopter Aerial Vehicle (QAV); Particle Swarm Optimization (PSO)

Received on 18 May 2022, accepted on 4 July 2022, published on 12 July 2022

doi: 10.4108/eetct.v9i31.1922

1. Introduction

Quadrotor is a type of helicopter that can be skillful by variable the rotor hurries. It is an under-actuated, lively vehicle with four input militaries and six output organizes. Quadrotor, calm of four rotors with proportionally preparation where two slanting motors (1 and 2) are consecutively in the same way while the others (3 and 4) in the other way to remove the anti-torque [1-5].The technique used in the quadcopter system, controller design, implementation of PSO algorithm[5-7], and procedure for achieving the best tuning. Also, the mathematical model and analysis of PSO algorithms are explained. These algorithms are used to improve the control of the quadcopter by applying them to tune the parameters of PIDC. Many factors are needed to operate a UAV[8]. The control is defined as the capacity to determine, maintain, or modify an airplane's attitude with respect to its intended flight path. A quadcopter with 6-DOF has fewer autonomous control inputs, making it difficult to retain control over all six. The axes, as well as yaw, pitch, and roll can now be controlled using the newly proposed design[9].

The trajectory planning is used to move the quadrotor from its initial location to the target position by determining the quadrotor's rotor velocities. Design and analysis of optimization process was performed to evaluate the vehicle's ideal path. The lodge of a system is necessary to specify the movement of the system, forces involved, and its response to the inputs [10], [11, 12].

[*]Corresponding author. Email: abdullah.algizi@stu.edu.iq

The goal is to demonstrate the simulation process of the quadcopter and introduce the control trajectory design used for the simulated flight path for this paper. The basic working principle of PIDC (Proportional, Integral, and Derivative) algorithm for quadcopter linear control systems are discussed in depth. The feedback linearization method for quadcopter attitude control was used because of its advantages and we then the outcomes were analyzed.

2. Simulation of The Quadcopter System

Simulink contained many points' path references of the simulation model wherein the waypoint follower block of Simulink's was utilized as the model. When input in MATLAB program it created a subdirectory in the project directory that contained the primary parameters and models required to simulate a flight route. This application may either simulate the intended flying route for the quadcopter using its 3D visualizer or it can begin flying and follow a pre-configured set of waypoints. Figure 1 illustrates the Simulink the quadcopter's initial flight model including the subsystem component blocks for waypoint commands, control system, plant model, and 3D flight display[13]. Figure 1 illustrates the Simulink the quadcopter's initial flight model including the subsystem blocks for waypoint commands, control system, plant model, and 3D flight display.

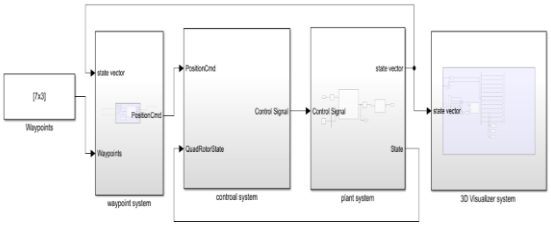


Figure 1. Simulink and simulation quadcopter model

2.1 Waypoint System

To start, the Simulink model mission requires the drone to reach the path points, while using a constant block to enter the desired path points (wps) into a waypoint follower block. matrix's waypoints show below in Figure 2. The waypoint (WP) follower, from the Simulink system UAV toolbox, calculates the target yaw, heading, and look-ahead point based on the quadcopter's position, WPs, and look-ahead distance. The desired WPs created for this study are entered into the WP box given by[14]:

$$wps = \begin{bmatrix} 0 & 100 & 150 & -100 & -150 & -50 & 100 & 0 \\ 0 & 100 & 150 & 100 & -150 & -50 & 100 & 0 \\ 100 & 100 & 100 & 100 & 100 & 100 & 100 & 100 \end{bmatrix} \quad (1)$$

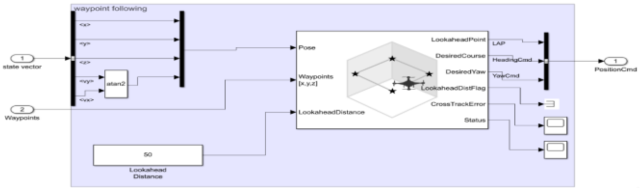
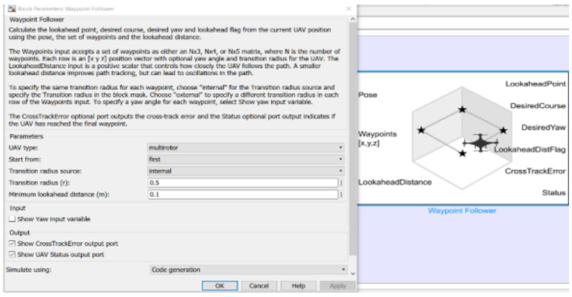


Figure 2. Waypoint subsystem design



Block Parameters: Waypoint Follower

Waypoint Follower

Calculate the lookahead point, desired course, desired yaw and lookahead flag from the current UAV position using the pose, the set of waypoints and the lookahead distance.

The Waypoints input accepts a set of waypoints as either an Nx3, Nx4, or Nx5 matrix, where N is the number of waypoints. Each row is an [x y z] position vector with optional yaw angle and transition radius for the UAV. The LookaheadDistance input is a positive scalar that controls how closely the UAV follows the path. A smaller lookahead distance improves path tracking, but can lead to oscillations in the path.

To specify the same transition radius for each waypoint, choose "internal" for the Transition radius source and specify the Transition radius in the block mask. Choose "external" to specify a different transition radius in each row of the Waypoints input. To specify a yaw angle for each waypoint, select Show yaw input variable.

The CrossTrackError optional port outputs the cross-track error and the Status optional port output indicates if the UAV has reached the final waypoint.

Parameters

UAV type: multirotor

Start from: first

Transition radius source: internal

Transition radius (r): 0.5

Minimum lookahead distance (m): 0.1

Input

□ Show Yaw input variable

Output

☑ Show CrossTrackError output port

☑ Show UAV Status output port

Simulate using: Code generation

Figure 3. Parameters used in waypoint follower block

The chosen WPs value in Equation 1 are intended to simulate all possible flying movements and rotations for a drone. Each WP has unique requirement for desired (position, yaw, pitch, and roll). This architecture indicated that all control commands implemented are applicable to any desired flying path. The order and direction of flight for the chosen WPs objective sites are shown in Figure 4

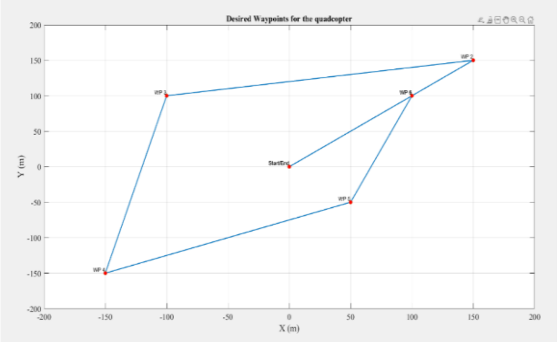


Figure 4. Waypoint flight path

2.2. Control System

The suggested control technique involved the combination of a position, velocity and an attitude controller. Due to the unreliability of open-loop control methods, a PIDC is used to regulate the quadrotor's closed-loop altitude. IAE is used in conjunction with the PSO method to avoid selecting gains that cause early mistakes high enough to make the job impossible. Additionally, disregard scenarios in which the UAV fails to reach the point, is too slow to reach the point, or reaches the target location but oscillates with an undesired amplitude. Gain difference intervals were determined based on the investigated system's behavior and, more specifically, the controller's responses in order to minimize saturation of the control signals[15].

As a result, a PID with chose PSO-gain is used to adjust the quadrotor's closed-loop fast controls to reach the desired point. PIDC is beneficial due to its simplicity and ease of implementation and takes the general mathematical form as in Equation 2,3 [15, 16]

$$e(t) = x_d(t) - x(t) \tag{2}$$

$$u(t) = k_p e(t) + k_i \int_0^t e(\tau)d\tau + k_d \frac{de(t)}{dt} \tag{3}$$

where u (t) is the control input, e(t) is the difference between the intended state x d(t) and current state x(t); KP, KI, and KD are the proportional, integral, and derivative components of PIDC, respectively. A quadrotor has three main subsystem blocks constituting the flight control system wherein the position controller is responsible for the variables *x*, *y*, *z* as shown in Figure 5[16].

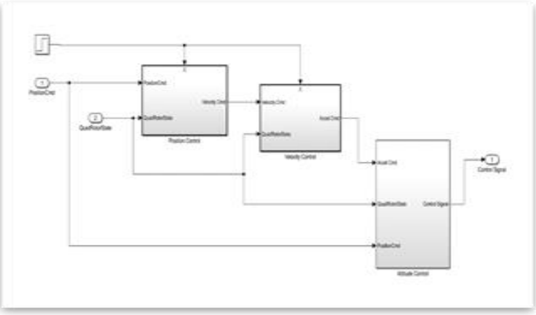
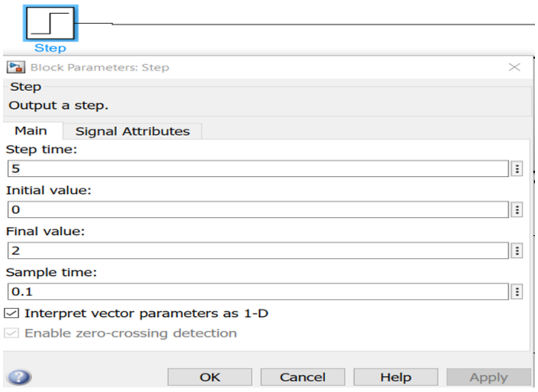


Figure 5. Control system design strategy



Step
Output a step.

Step time:
5

Initial value:
0

Final value:
2

Sample time:
0.1

☑ Interpret vector parameters as 1-D
☑ Enable zero-crossing detection

Figure 6. The parameter of step block

2.3 Position And Velocity Controllers

The position control is a method for keeping track of moving vehicles using their current locations. In the position and velocity controls, there are three PID blocks that drive the distance to velocity translation as displayed in Figure 7. The derivation of the vehicle speed gives the acceleration of the vehicle Figure 8 via the equations 5-7:[17].

$$v_x = -k_1^x e_x - k_2^x \dot{e}_x \tag{5}$$

$$v_y = -k_1^y e_y - k_2^y \dot{e}_y \tag{6}$$

$$v_z = -k_1^z e_z - k_2^z \dot{e}_z \tag{7}$$

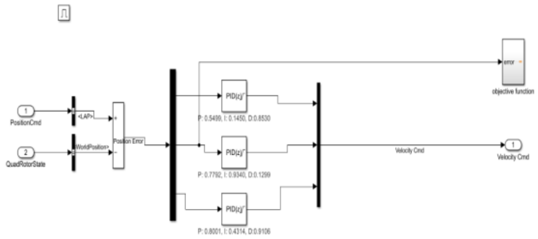


Figure 7. Position control design strategy

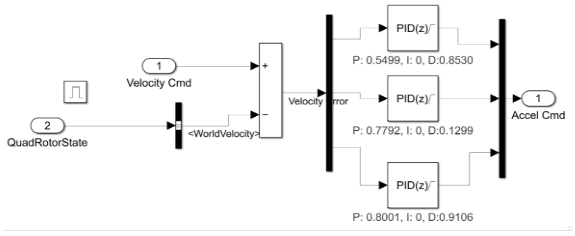


Figure 8. Velocity control design strategy

2.4 The attitude controller

PIDCs are essential to tackle the problem and differences between their usages as seen clearly from the output. The tuning of the PIDC parameters for both roll angle, pitch angle and yaw angle loops can be accomplished by adjusting the PID coefficients. The proportional gain coefficient is used to manage acceleration and stability, while the integral coefficient is used to reduce permanent fault, and lastly, the derivative coefficient is used to reduce oscillation range and increase stability. The altitude controller was developed to follow a trajectory path using the Simulink block diagram as shown in Figure 9.[17, 18].

By applying the force and moment balance laws, the Quadcopter motion equation are given in Equation (8) till (11)[19].

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = -g \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \frac{F_T}{m} \begin{bmatrix} cos\Psi sin\theta cos\varphi + sin\psi sin\varphi \\ sin\psi sin\theta cos\varphi - cos\psi sin\varphi \\ cos\theta cos\varphi \end{bmatrix} \tag{8}$$

$$pitch = \tan^{-1} \frac{(-\ddot{x}\cos\psi - \ddot{y}\sin\psi)}{-\ddot{z}+g} \tag{9}$$

$$roll = \tan^{-1} \frac{cos\theta * (tan\theta * sin\psi + \frac{\ddot{y}}{-\ddot{z}+g})}{sin\psi} \tag{10}$$

$$\text{thrust} = \text{droneMass} * \frac{(-\ddot{z}+g)}{\cos\theta \cos\phi} \tag{11}$$

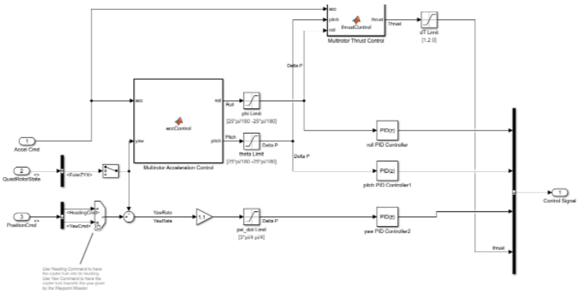


Figure 9. Altitude control design strategy

3. Quadcopter Plant

The quadrotor's governing equations were used to develop the system's dynamic and kinematics model. First, the blocks of the system received the required trajectory and passed the model's ideal state to the state estimation output vectors. Next, the state estimation function determined the current state of the model by comparing it to the model's state. The plant determines the current state in the body frame of the model by comparing it to the desired path. The controller will at first receive a large error, and the control signal should be high. As the robot gets closer to the desired point, the control signal will get smaller, tending toward zero, wherein the quadrotor should stabilize until the position and velocity errors are zero. Figure 10 shows the basic architecture of Simulink and simulation of quadcopter plant. Figure 11 displays the parameters of quadcopter plant. The quadcopter will start from its initial position (0, 0, 0) [m] and thrust value 1 N as see in Fig. 12.

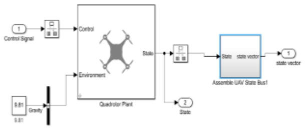


Figure 10. Basic design of Simulink and simulation of quadcopter plant.

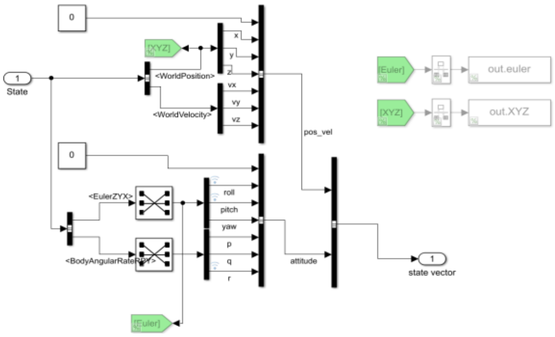
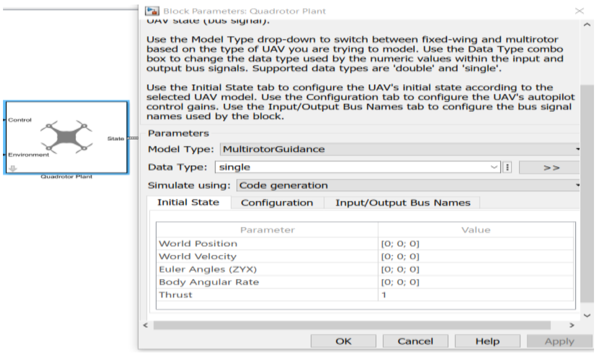


Figure 11. Basic architecture of Simulink and simulation quadcopter plant



UAV state (bus signal).

Use the Model Type drop-down to switch between fixed-wing and multirotor based on the type of UAV you are trying to model. Use the Data Type combo box to change the data type used by the numeric values within the input and output bus signals. Supported data types are 'double' and 'single'.

Use the Initial State tab to configure the UAV's initial state according to the selected UAV model. Use the Configuration tab to configure the UAV's autopilot control gains. Use the Input/Output Bus Names tab to configure the bus signal names used by the block.

Parameters

Model Type: MultirotorGuidance

Data Type: single

Simulate using: Code generation

Initial State | Configuration | Input/Output Bus Names

Parameter	Value
World Position	[0; 0; 0]
World Velocity	[0; 0; 0]
Euler Angles (ZYX)	[0; 0; 0]
Body Angular Rate	[0; 0; 0]
Thrust	1

Figure 12. Parameter of quadcopter plant

3.1 D Visualizer System

The UAV animation block animated a quadcopter flight path based on an input array of translations and rotations.

In rotations port, using the coordinate transformation, the conversion block makes a coordinate transformation from the input representation to a specified output representation, displaying the visual mesh for multi-rotor at the given position and orientation. By clicking the show animation button in the block mask, it is possible to bring up the figure after simulating. The final view is the QAV view that allows for a top-down tracking while the drone is in flight show in Figure 13. Figure 14 illustrates the parameters of UAV animation block.

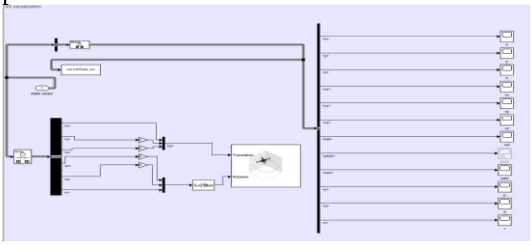
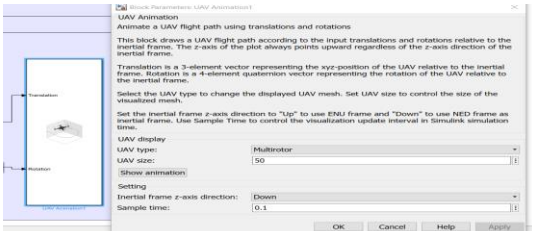


Figure 13. 3D Visualizer system



Block Parameters: UAV Animation1

UAV Animation

Animate a UAV flight path using translations and rotations

This block draws a UAV flight path according to the input translations and rotations relative to the inertial frame. The z-axis of the plot always points upward regardless of the z-axis direction of the inertial frame.

Translation is a 3-element vector representing the xyz-position of the UAV relative to the inertial frame. Rotation is a 4-element quaternion vector representing the rotation of the UAV relative to the inertial frame.

Select the UAV type to change the displayed UAV mesh. Set UAV size to control the size of the visualized mesh.

Set the inertial frame z-axis direction to "Up" to use ENU frame and "Down" to use NED frame as inertial frame. Use Sample Time to control the visualization update interval in Simulink simulation time.

UAV display

UAV type: Multirotor

UAV size: 50

Setting

Inertial frame z-axis direction: Down

Sample time: 0.1

Figure 14. The parameter of UAV animation block

3.2 Particle Swarm Optimization

Following the swarm intelligence technology introduced by Eberhart and Kennedy, the optimization problems can be solved using PSO. PSO takes its cues from animal groups such as birds, fish, and herds of animals are all examples of flocks. All of which have evolved an evolutionary advantage through the use of "information sharing" systems. There are several iterations of propagation in which a randomly produced population of particles (called an "initial swarm") populates the PSO and a random velocity is assigned to all of those particles. A particle's "personal best position" is a memory of the highest position it has ever achieved (Post). It is the particle with the highest P best that is referred to as the "global best particle"[20, 21].

Let us consider that the search space is D-dimensional wherein a D-dimensional vector $(x_i^1, x_i^2, \dots, x_i^D)^T$ represents the ith member of the population. These moves at a higher speed and the velocity represent another particle in the D-dimensional space through another vector $(V_i^1, V_i^2, \dots, V_i^D)^T$. Pi denotes the particle's previous best position, while the best particle is denoted by Pi and Pg denotes the presence of the swarm. The following two equations are used to update the particle's position. After calculating the previous velocity, Eq. (12) updates each particle's position in the search space by computing a new velocity for each particle Next, Eq. (13) updates each particle’s position in search space[22, 23].

$$V_{id}^{k+1} = wV_{id}^{k} + c_1 r_1 [p_{id}^{k} - x_{id}(t)] + c_2 r_2 [p_g^k(t) - x^k{}_{id}(t)] \tag{12}$$

$$[x^{k+1}{}_{id}(t+1)] = x^k{}_{id}(t) + V_{id}^{k+1}(t+1) \tag{13}$$

where k¼ is the iteration number; d¼ can take values 1,2,3, …, D; i¼ can take values 1,2,3, …, N; N¼ is the swarm size; w¼ is the inertia weight that controls the momentum of particle by weighing the contribution of previous velocity; c1 and c2 are the positive constants called acceleration coefficients; r1 and r2 are the random numbers uniformly distributed between [0,1] [24, 25].

3.3 Objective Function for tuning PID control

The importance of optimization parameters or the results of an operation used as input to the optimization algorithm is described mathematically as the objective function. Figure 15 shows the fitness function for tuning PID controls. The PID controller is employed with all optimization techniques to obtain the lowest feasible error and to offer the PID controller's overall magnitude as a preferred alternative [53] The PID tuning values were mentioned in Appendix A. Objective function calculate from the following Integral absolute error Equation in (14)-(15)[26].

The objective function can be calculated in terms of the integral absolute error (IAE) or performance index via:

$$\text{Objective function} = \frac{1}{\text{performance index}} \tag{14}$$

$$\text{performance index} = \text{IAE} = \int_0^\infty |u| \, dt \tag{15}$$

IAE: Integral absolute error

u= The desired angle - The actual angle u= error

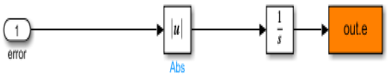


Figure 15. Fitness function for tuning PID control

3.4 Simulation Results

The QAV is flown in a detailed order to six chosen WP. The three-dimensional image demonstrates that the gain values have been set appropriately; The grid spacing represents one meter. Throughout the scenario, the QAV begins at a distance of one meter. This is a setting within the plant model that moves vertically first and then angularly to accomplish the first WP. This arises as a result of the plant's basic settings. The following WPs are accomplished by establishing the appropriate heading and yaw and then proceeding at a determined speed to a specified distance[1, 2, 27-30].   Figure 16 displays the 3d view of the simulated flight trajectory of the QAV wherein it is flown in a detailed order to six chosen WP. The obtained 3D image demonstrates that the gain values have been set appropriately. The grid spacing represents one meter and throughout the scenario, the QAV began at a distance of one meter. This is a setting within the plant model that moves vertically first and then angularly to accomplish the first WP. This arises as a result of the plant's basic settings. The following WPs are accomplished by establishing the appropriate heading and yaw and then proceeding at a determined speed to a specified distance. Figure 17 shows the 2D view of Simulink flight path of the QAV display like simulation waypoint. Thus, the flight path confirmed that the QAV made the most efficient route to the targeted WP. The results simulated flight trajectory of the QAV is presented from Figures 18- 29. These figures represent the results of the following equations:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{bmatrix} = R\begin{bmatrix}u\\ v\\ w\end{bmatrix} \quad , \quad \begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = J\begin{bmatrix}p\\ q\\ r\end{bmatrix} \qquad \text{-- (16)}$$

As a result, final state vector of quadcopter is:

$$x(t) = \begin{bmatrix}P\\ V\\ \Phi\\ \Omega\end{bmatrix} = \begin{bmatrix}x\\ y\\ z\\ u\\ v\\ w\\ \varphi\\ \theta\\ \psi\\ p\\ q\\ r\end{bmatrix} \qquad \text{equation derivation in - (17)}$$

Were P: position in earth frame, V: velocity in earth frame, Φ :angles of axis in earth frame, Ω : angular velocity in body frame

the quadcopter's output state vector in Equation 17 continues to track the six degrees of freedom with extremely minimal errors. When the x, y, and z reference distance values are tracked, a significant overshoot is noted at the start of the test period. This is because the quadcopter begins its journey from a point much outside the trajectory. However, once it reaches the desired route, the quadcopter's velocities stabilize and track the reference values quite smoothly.

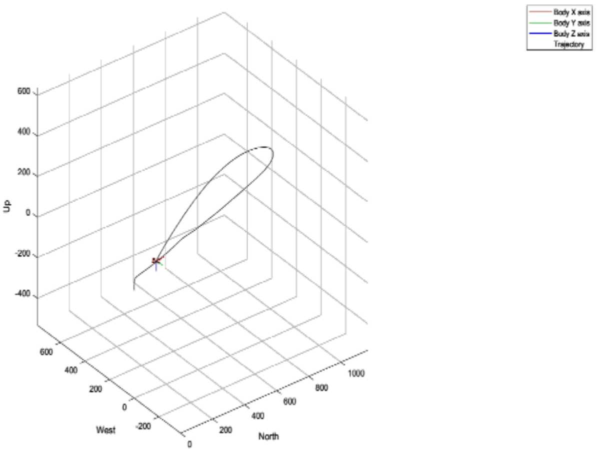


Figure 16. 3D view of Simulink flight path of the QAV

To better imagine the flight path taken 2D by the Simulink, a plot concerning the position in the x, z and y, z axes are displayed in Figure 17,18 and 19, respectively. Figures 20, 21 and 22 presents the X, Y and Z distance response of the QAV, respectively. Figures 23, 24 and 25 illustrate the corresponding linear velocity function of QAV along X, Y and Z direction with time. Figures 26 ,27,28,29,30 and 31 display the angle around X-, Y- and Z-axes of the QAV, respectively. Figures 28, 29 and 30 depict the angular velocity of QAV around X-, Y- and Z-axes, respectively.

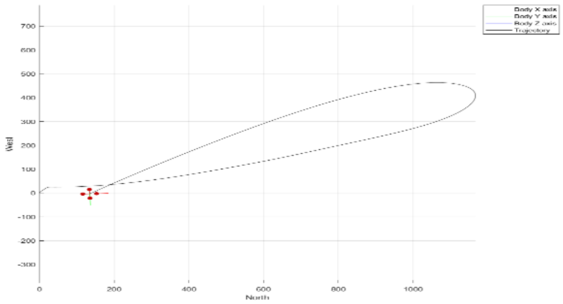


Figure 17. 2D (X-Y) view of Simulink Flight path of the QAV

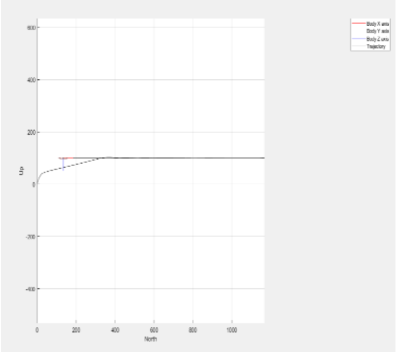


Figure 18. 2D (X-Z) view of Simulink Flight path of the QAV

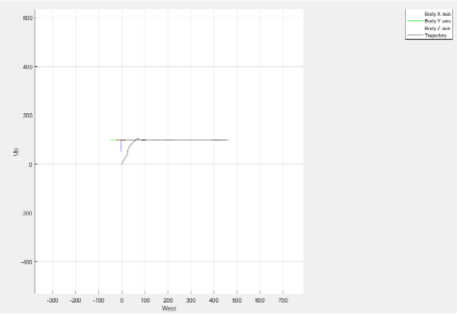


Figure 19. 2D (Y-Z) view of Simulink flight path of the QAV

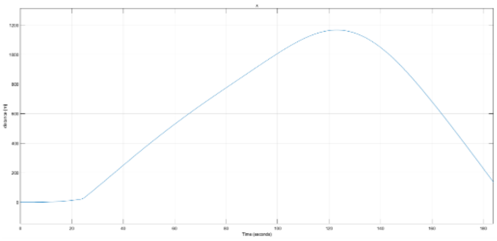


Figure 20. X distance response for the QAV

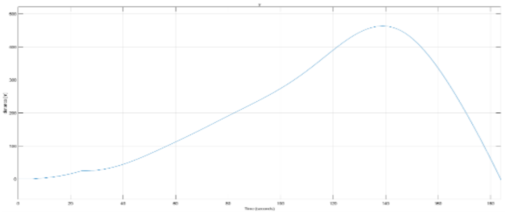


Figure 21. Y distance response for the QAV

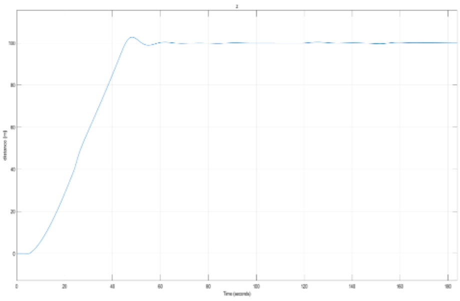


Figure 22. Z distance response for the QAV

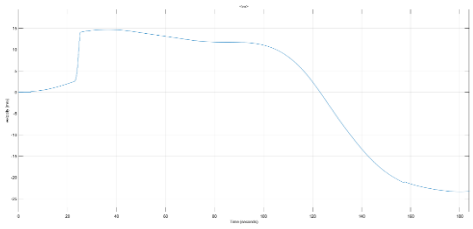


Figure 23. U linear velocity response for the QAV

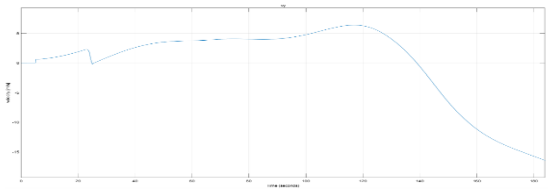


Figure 24. V linear velocity response for the QAV

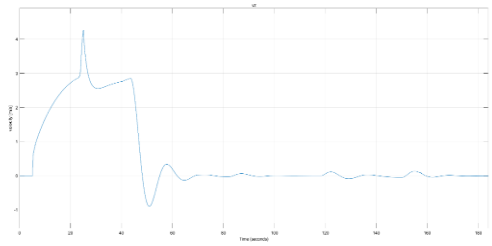


Figure 25. W linear velocity function with time for the QAV

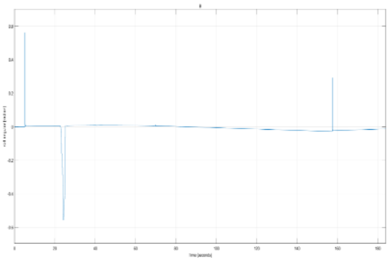


Figure 26. ϕ angle around X - axes for the QAV

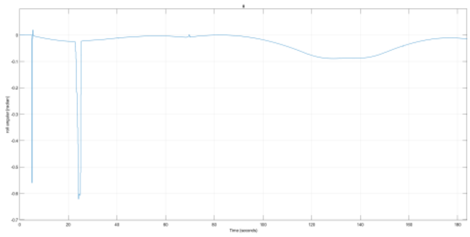


Figure 27. θ angle around Y- axes for the QAV

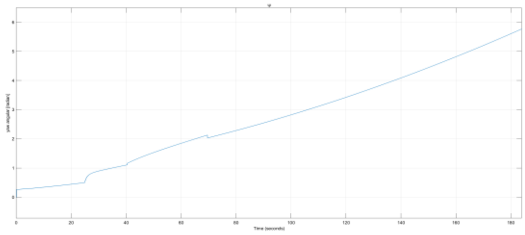


Figure 28. ψ angle around z - axes for the QAV

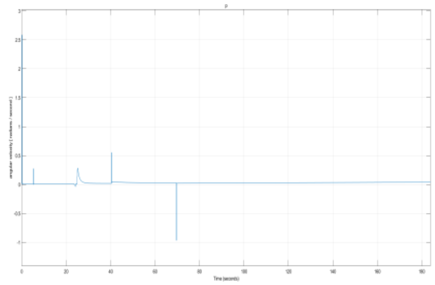


Figure 29. P angular velocity around X - axes for the QAV

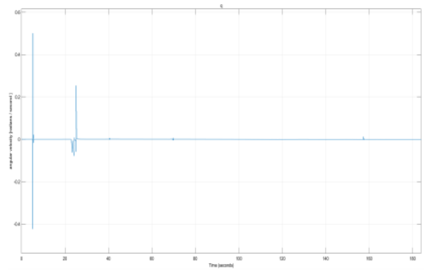


Figure 30. Q angular around Y - axes for the QAV

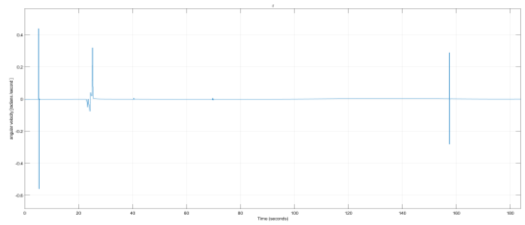


Figure 31. R angular velocity around Z - axes for the QAV

Figure 32 illustrates the 3D view of Simulink Flight path without attitude controller. Figures 33- 36 show the plot of various output state vectors of the quadcopter as equation 17 continued to track the six degrees of freedom with lowest possible errors. A significant overshoot was noted at the start of the test period when x, y, and z reference distances were tracked. This observation was ascribed to the starting journey of the quadcopter from a point far away from the trajectory. However, the velocities of the quadcopter were stabilized and the reference values were tracked quite smoothly once it reached the desired route. It was asserted that the Newton-Euler kinematic model can be used to simulate the quadcopter's altitude during a trajectory flight. In addition, the quadcopter must have a control system for continuous movement in order to remain stable during the motions with constantly varying yaw, pitch, and roll angles. The simulation results showed a significant overshoot without any control mechanism in place.

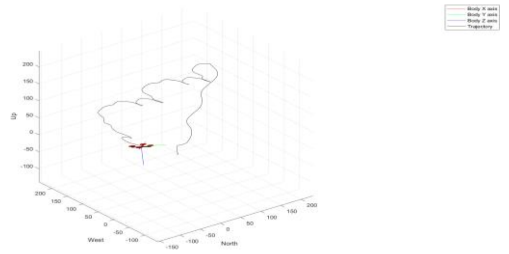


Figure 32. 3D view of Simulink flight path without attitude controller

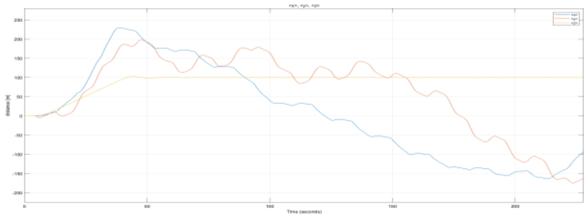


Figure 33. Plot distance P vector without attitude controller

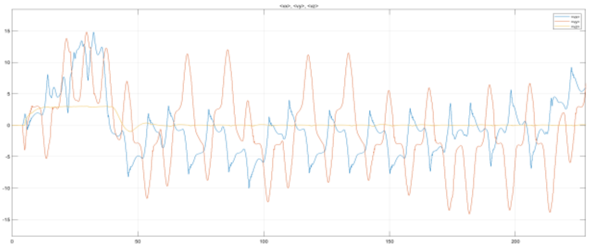


Figure 34. Plot linear velocity vector without attitude controller

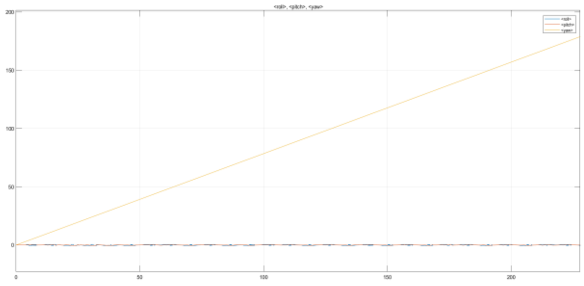


Figure 35. Plot angle vector without attitude controller

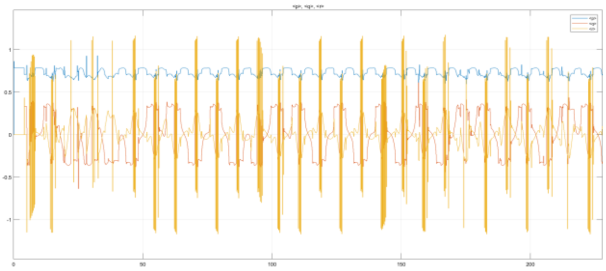


Figure 36. Plot angular velocity vector without attitude controller

4. Conclusion

This chapter demonstrated that the quadcopter’s altitude movement on a trajectory flight can be modelled using the Newton-Euler kinematic model. A continuous movement control system is required to ensure the stability of the quadcopter while navigating that involves continuous changes in the yaw, pitch, and roll angles. Without a control system, significant amounts of overshoot can readily be produced in the horizontal plane of altitude movement and the simulated results show the aircraft moving in an unstable manner, deviating from the needed course, and failing to return to the original point. Thus, to reduce the inaccuracy of flight, PIDC can be added to the quadcopter movement model. Only proportional and derivative components are required to guarantee the quadcopter flying position stability in the case of the PID control system. This clearly suggested that the quadcopter continues to fly on a trajectory with minimum inaccuracy, wherein the PIDC operated as a closed-loop adaptive controller. Finally, the PSO-PIDC outperformed the PIDC due to the critical nature of manual gain modification for quadcopter flight stability.

Acknowledgements.

Special thanking to **Saja Raad Abdulbari Hani** Saja.alhilfi@gmail.com& prof Sib K.G mailto:sibkrishna@utm.my for critical readying.

References

[1] S. H. Derrouaoui, Y. Bouzid, and M. Guiatni, "PSO Based Optimal Gain Scheduling Backstepping Flight Controller Design for a Transformable Quadrotor," Journal of Intelligent & Robotic Systems, vol. 102, no. 3, pp. 1-25, 2021.

[2] V. P. Tran, F. Santoso, and M. A. Garratt, "Adaptive trajectory tracking for quadrotor systems in unknown wind environments using particle swarm optimization-based strictly negative imaginary controllers," IEEE Transactions on Aerospace and Electronic Systems, vol. 57, no. 3, pp. 1742-1752, 2021.

[3] E. Camci, D. R. Kripalani, L. Ma, E. Kayacan, and M. A. Khanesar, "An aerial robot for rice farm quality inspection with type-2 fuzzy neural networks tuned by particle swarm optimization-sliding mode control hybrid algorithm," Swarm and evolutionary computation, vol. 41, pp. 1-8, 2018.

[4] V. Gomez, N. Gomez, J. Rodas, E. Paiva, M. Saad, and R. Gregor, "Pareto optimal PID tuning for Px4-based unmanned aerial vehicles by using a multi-objective particle swarm optimization algorithm," Aerospace, vol. 7, no. 6, p. 71, 2020.

[5] I. M. Lazim, A. R. Husain, N. A. Mohd Subha, Z. Mohamed, and M. A. Mohd Basri, "Optimal formation control of multiple quadrotors based on particle swarm optimization," in Asian Simulation Conference, 2017: Springer, pp. 121-135.

[6] H.-K. Tran and J.-S. Chiou, "PSO-based algorithm applied to quadcopter micro air vehicle controller design," Micromachines, vol. 7, no. 9, p. 168, 2016.

[7] B. N. A. Samed and A. A. Aldair, "Design Tunable Robust Controllers for Unmanned Aerial Vehicle Based on Particle Swarm Optimization Algorithm," Iraqi Journal for Electrical & Electronic Engineering, vol. 15, no. 2, 2019.

[8] T. He, Y. Zeng, and Z. Hu, "Research of multi-rotor UAVs detailed autonomous inspection technology of transmission lines based on route planning," IEEE Access, vol. 7, pp. 114955-114965, 2019.

[9] T. K. Priyambodo, A. Dharmawan, O. A. Dhewa, and N. A. S. Putro, "Optimizing control based on fine tune PID using ant colony logic for vertical moving control of UAV system," in AIP Conference Proceedings, 2016, vol. 1755, no. 1: AIP Publishing LLC, p. 170011.

[10] A. Reizenstein, "Position and Trajectory Control of a Quadcopter Using PID and LQ Controllers," ed, 2017.

[11] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," The International Journal of Robotics Research, vol. 31, no. 5, pp. 664-674, 2012.

[12] N. H. Abbas and A. R. Sami, "Tuning of PID controllers for quadcopter system using hybrid memory based gravitational search algorithm–particle swarm optimization," International Journal of Computer Applications, vol. 172, no. 04, pp. 9-18, 2017.

[13] E. Kuantama, T. Vesselenyi, S. Dzitac, and R. Tarca, "PID and Fuzzy-PID control model for quadcopter attitude with disturbance parameter," International journal of computers communications & control, vol. 12, no. 4, pp. 519-532, 2017.

[14] G. Farid, H. T. Hamid, S. Karim, and S. Tahir, "Waypoint-based generation of guided and optimal trajectories for autonomous tracking using a quadrotor uav," Studies in Informatics and Control, vol. 27, no. 2, pp. 225-236, 2018.

[15] L. Fagundes-Junior, M. Canesche, R. Ferreira, and A. Brandão, "A Nonlinear UAV Control Tuning Under Communication Delay using HPC Strategies in Parameters Space," in Anais do XXII Simpósio em Sistemas Computacionais de Alto Desempenho, 2021: SBC, pp. 228-239.

[16] A. Goel, A. M. Salim, A. Ansari, S. Ravela, and D. Bernstein, "Adaptive digital pid control of a quadcopter with unknown dynamics," arXiv preprint arXiv:2006.00416, 2020.

[17] M. A. Khodja, M. Tadjine, M. S. Boucherit, and M. Benzaoui, "Tuning PID attitude stabilization of a quadrotor using particle swarm optimization (experimental)," International Journal for Simulation and Multidisciplinary Design Optimization, vol. 8, p. A8, 2017.

[18] R. Thusoo, S. Jain, and S. Bangia, "PID Control of a Quadrotor," in Advances in Communication and Computational Technology: Springer, 2021, pp. 633-645.

[19] A. Z. Ruwodo, "Dynamic quadrotor traffic offense monitoring system," 2018.

[20] X. Wang and Y. Li, "Chaotic image encryption algorithm based on hybrid multi-objective particle swarm optimization and DNA sequence," Optics and Lasers in Engineering, vol. 137, p. 106393, 2021.

[21] A. Kumar, S. Pant, M. Ram, and S. Singh, "On solving complex reliability optimization problem using multi-objective particle swarm optimization," in Mathematics Applied to Engineering: Elsevier, 2017, pp. 115-131.

[22] S. Saeedi, R. Khorsand, S. G. Bidgoli, and M. Ramezanpour, "Improved many-objective particle swarm optimization algorithm for scientific workflow scheduling in cloud computing," Computers & Industrial Engineering, vol. 147, p. 106649, 2020.

[23] M. Abdalla and S. Al-Baradie, "Real Time Optimal Tuning of Quadcopter Attitude Controller Using Particle Swarm Optimization," J. Eng. Technol. Sci, vol. 52, pp. 745-764, 2020.

[24] H.-H. Tsai, C.-C. Fuh, J.-R. Ho, and C.-K. Lin, "Design of Optimal Controllers for Unknown Dynamic Systems through the Nelder–Mead Simplex Method," Mathematics, vol. 9, no. 16, p. 2013, 2021.

[25] A. Taeib, A. Ltaeif, and A. Chaari, "A PSO Approach for Optimum Design of Multivariable PID Controller for nonlinear systems," arXiv preprint arXiv:1306.6194, 2013.

[26] J. Connor, M. Seyedmahmoudian, and B. Horan, "Using particle swarm optimization for PID optimization for altitude control on a quadrotor," in 2017 Australasian Universities Power Engineering Conference (AUPEC), 2017: IEEE, pp. 1-6.

[27] N. Sun, T. Yang, Y. Fang, Y. Wu, and H. Chen, "Transportation control of double-pendulum cranes with a nonlinear quasi-PID scheme: Design and experiments," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 49, no. 7, pp. 1408-1418, 2018.

[28] H. Litimein, Z.-Y. Huang, and A. Hamza, "A Survey on Techniques in the Circular Formation of Multi-Agent Systems," Electronics, vol. 10, no. 23, p. 2959, 2021.

[29] C. Xu, X. Liao, J. Tan, H. Ye, and H. Lu, "Recent research progress of unmanned aerial vehicle regulation policies and technologies in urban low altitude," IEEE Access, vol. 8, pp. 74175-74194, 2020.

[30] B.-M. Nguyen, T. Kobayashi, K. Sekitani, M. Kawanishi, and T. Narikiyo, "Altitude Control of Quadcopters with Absolute Stability Analysis," IEEJ Journal of Industry Applications, p. 21010314, 2022.