FAZ Publishing
JCF

Journal of Complex Flow, Vol. 2 No. 2 (2020) p. 19-26

Journal of Complex Flow

Journal homepage : www.fazpublishing.com/jcf

e-ISSN : 2672-7374

Dynamic Modelling and Control for Quadcopter UAV with LabVIEW and X-Plane Flight Simulator

Yusuf Ali Ahmad Zabidin[1], Mohammad Fahmi Pairan[1], Syariful Syafiq Shamsudin[1,*],

[1]Faculty of Mechanical and Manufacturing Engineering,
Universiti Tun Hussein Onn Malaysia, Batu Pahat, 86400, MALAYSIA

*Corresponding Author

Email: syafiq@uthm.edu.my

Received 12 June 2020;
Accepted 17 September 2020;
Available online 15 October 2020

Abstract: This study presents a modeling and control method for the quadcopter based unmanned aerial vehicle (UAV) using the software-in-the-loop (SITL) approach. The proposed SITL approach is used as a simulation tool to validate the control techniques applied to the quadcopter UAV before the actual flight test. The dynamics model of quadcopter UAV is implemented in the X-Plane flight simulator and the automatic flight controller is implemented in National Instruments LabVIEW software. The quadcopter has four actuators with fewer control inputs than the degrees of freedom, complicating its control mechanism. The controlled motions of the quadcopter are its forward, lateral and vertical motions along with its heading rate and attitude (yaw, pitch, and roll). The controller developed for the quadcopter under consideration is based on the cascaded PID controllers. In this approach, the quadcopter dynamic is decomposed into two cascaded loops (i.e., inner and outer loop controls). The cascaded PID controllers are implemented in LabVIEW and connected to the X-Plane flight simulator using the UDP data interface. The tuning of the PID controller is done manually and the quadcopter responses are recorded.

Keywords: Flight simulator; PID controller; Quadcopter; Software-in-the-loop method; Simulation; Unmanned aerial vehicle

1. Introduction

A quadcopter is categorized as a helicopter that has four rotors arranged in symmetry around a central body, usually square-shaped. The rotors are individually powered while adjustments in the angular speed of the rotors provide control over the quadcopter [1]. Unmanned Aerial Vehicle (UAV) is an aircraft that is controlled without a pilot on board the aircraft. The UAV can be operated remotely or autonomously. Quadcopter UAVs, colloquially known as 'drones', are UAVs designed in the form of a quadcopter. They are used in many fields, both civilian and military, in applications such as surveillance and inspections.

The control of four independent rotors of a quadcopter is difficult and complicated and requires major electronic assistance. Advances in electronics have made modern processors and sensors smaller and cheaper, making it possible for quadcopters to be as small as a thumb [2] and cheap enough that quadcopters can be homemade. The challenge of controlling quadcopters comes from quadcopters having four actuators in the form of four separate rotor velocities with six degrees of freedom. The six degrees of freedom is obtained by coupling the motions of translation and rotation, thereby producing highly nonlinear dynamics [3]. The different conditions in the flight of quadcopters made it so that the aerodynamic forces acting upon the quadcopter are highly variable, resulting in the complexity of controlling the quadcopter. Researches into different control methods include PID controllers, back-stepping control, nonlinear H∞ controls, linear quadratic LQR algorithms, and feedback linearization [3]. The quadcopter control method focused on in this study is the PID controller method.

Proportional-Integral-Derivative (PID) controller is widely used as a closed-loop control mechanism which uses proportional K_p, integral K_i or derivative K_d, gain terms to compute the desired actuator output. A PID controller constantly calculates error value between the desired set point of a variable and the measured process variable and applies a correction based on the PID controller gains. Most commercially available quadcopters use PID controllers in their autopilot system.

**Corresponding author: syafiq@uthm.edu.my*

The tuning of PID gain values is very much a requirement to achieve the optimal gain values for the response desired from the control system. In quadcopters, the complexity of its control requires tuning the PID gain values so that the flight performance of the quadcopter can be improved. Stability, reliable control and improved handling are some of the main reasons for tuning. The commonly used method of tuning the PID controller of a quadcopter is the trial and error method by which the controller gain values are adjusted, and then the quadcopter is flown using the newly adjusted controller gain values. This method is highly tedious as for each time an adjustment in the controller gain value is made. This method also makes the quadcopter prone to collisions due to suboptimal controller gain values causing poor controller response in flight.

The goals of this research are to develop a linear dynamic model and control simulation of a quadcopter to assist in the PID tuning process. The performance of the developed control algorithm is evaluated in hovering and low-speed flight tests. The study used the National Instruments LabVIEW graphical programming approach for the controller and the simulator X-Plane software to simulate the hover and low-speed flight conditions.

2. Modelling and Methodology

This study uses the software-in-the-loop method in which the model of the quadcopter is implemented in X-Plane flight simulator while the controller is implemented in LabVIEW. The development of the quadcopter model began by the measurement of the moments of inertia of the quadcopter. Development of a model is done in National Instruments LabVIEW software using the linearized Newton-Euler flight dynamics equations. The model developed is then simulated in X-Plane software, a commercially available simulator. A control algorithm design for PID tuning is then developed using simulations based on the model. Once a suitable algorithm has been developed, the controller designed was implemented on the hardware (quadcopter) in the simulator. Flight tests were done in the simulation environment to validate the developed PID tuning algorithm. The flight tests simulated were for hover and low-speed flight performance.

The accurate modelling of the quadcopter requires measurements of the quadcopter's parameters and understanding the relations to the quadcopter's dynamics. The important parameters that were needed to be measured are the quadcopter's second moment of inertia and the thrust and torque output of the electric motors. The design tool Plane-Maker software associated with the X-Plane simulator was used to create a model of the quadcopter.

2.1 Quadcopter Dynamics

Quadcopters have six degrees of freedom but only have four actuators. This feature makes quadcopters an under-actuated and dynamically unstable system. The following are a summary of modelling equations used for the kinematics and dynamics of a quadcopter[4, 5].

The basic assumptions are that the structure of the quadcopter is rigid and symmetrical, and the center of gravity is coincident to the body-fixed frame origin. The following are the Newton-Euler equations developed for flight dynamics of a quadcopter.

$$\begin{pmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{h} \end{pmatrix} = \begin{pmatrix} A & B & C \\ D & E & F \\ G & H & I \end{pmatrix} \begin{pmatrix} \mu \\ \nu \\ w \end{pmatrix} \tag{1}$$

where,

$A = c\theta c\psi$ $\quad F = c\phi s\theta c\psi - s\phi s\psi$

$B = s\phi s\theta c\psi - c\phi s\psi$ $\quad G = s\theta$

$C = c\phi s\theta c\psi + s\phi s\psi$ $\quad H = -s\phi c\theta$

$D = c\theta c\psi$ $\quad I = -c\phi c\theta$

$E = s\phi s\theta c\psi + c\phi s\psi$

$$\begin{pmatrix} \dot{\mu} \\ \dot{\nu} \\ \dot{w} \end{pmatrix} = \begin{pmatrix} rv - qw \\ pw - ru \\ qu - pv \end{pmatrix} + \begin{pmatrix} -mg\ s\theta \\ mg\ c\theta\ s\phi \\ mg\ c\theta\ c\phi \end{pmatrix} + \frac{1}{m} \begin{pmatrix} 0 \\ 0 \\ -F \end{pmatrix} \tag{2}$$

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & s\phi t\theta & s\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi \sec\theta & c\phi \sec\theta \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix} \tag{3}$$

$$\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \begin{pmatrix} \frac{J_y - J_z}{J_x} qr \\ \frac{J_z - J_x}{J_y} pr \\ \frac{J_x - J_y}{J_z} pq \end{pmatrix} + \begin{pmatrix} \frac{1}{J_x} \tau_\phi \\ \frac{1}{J_y} \tau_\theta \\ \frac{1}{J_z} \tau_\psi \end{pmatrix} \tag{4}$$

where, $c\theta$, $s\theta$ and $t\theta$ represent notation for $\cos\theta$, $\sin\theta$ and $\tan\theta$. p_n and p_e are inertial north and inertial east position of the UAV. The altitude of the UAV also known as the inertial down position is indicate as h. The angular rates are noted by p for roll rate, q for pitch rate, and r for yaw rate. F denotes the force generated by all four quadcopter motors while m denotes the quadcopter mass. The moments of inertia are represented by I_{xx} for the moment of inertia about the x-axis, I_{yy} for y-axis moment, and I_{zz} for z-axis. Roll torque is indicated by τ_ϕ, τ_θ for pitch torque, and τ_ψ yaw torque with g for the standard gravity constant (9.81 m/s^2). The state variables of an aircraft are shown Fig. 1. The velocities of an aircraft are denoted by u for forward velocity, v for lateral velocity, and w for vertical velocity. Fig. 2 shows the torque directions and motor positions of a quadrotor as viewed from the top of the quadrotor. Fig. 3 shows the definition of forces acting on the quadrotor.

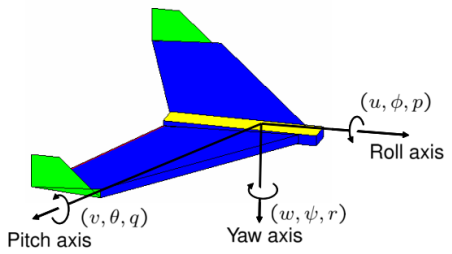

Fig. 1 - Schematic view of the state variables of an aircraft [4]

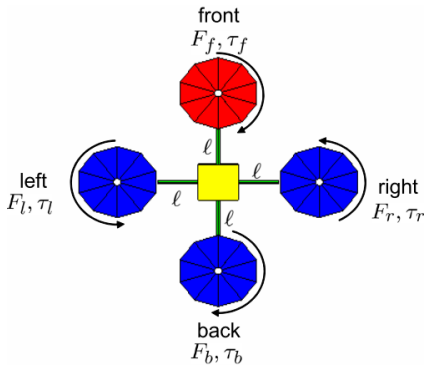

Fig. 2 - Top view of a quadrotor denoting the torque directions and the motor positions [4]

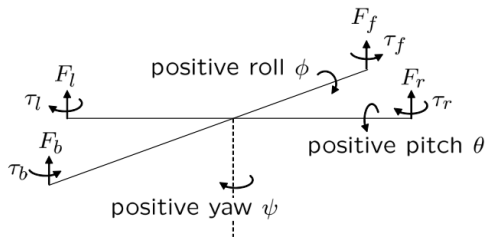

Fig. 3 - Definition of forces acting on the quadrotor [4]

The forces and torques acting on the quadcopter are primarily due to gravity and the four propellers as shown in Fig. 3. The equations for the total force acting on the quadrotor, F, rolling torque, τ_ϕ, pitching torque, τ_θ, and total yawing torque, τ_ψ, is as shown below [3][4]:

$$F = F_f + F_r + F_b + F_l \tag{5}$$

$$\tau_\phi = l(F_l - F_r) \tag{6}$$

$$\tau_\theta = l(F_f - F_b) \tag{7}$$

$$\tau_\psi = \tau_r + \tau_l + \tau_f + \tau_b \tag{8}$$

The gravitational force acting on the center of mass is given by,

$$f_g^b = R^T \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix} = \begin{pmatrix} -mg\, s\theta \\ mg\, c\theta\, s\phi \\ mg\, c\theta\, c\phi \end{pmatrix} \tag{9}$$

where R^T is the transposed rotational matrix representing the rotation from the body frame to inertial frame [3] which is given by,

$$R = \begin{pmatrix} A & B & C \\ D & E & F \\ G & H & I \end{pmatrix} \tag{10}$$

where A to I are similar as in Equation 1.

The lift and drag forces produced by the rotors is proportional to the square of the angular velocity of the rotors. The assumption is that the angular velocity is directly proportional to the pulse width modulation command sent to the motor. The force and torque of each motor can then be expressed as:

$$F_* = k_1 \delta_* \tag{11}$$

$$\tau_* = k_2 \delta_* \tag{12}$$

where k_1 and k_2 are experimentally determined constants, δ_* is the motor command signal, and * representing *f*, *r*, *b*, and *l*. Thus, the forces and torques acting on the quadrotor can be written as,

$$\begin{pmatrix} F \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{pmatrix} = \begin{pmatrix} k_1 & k_1 & k_1 & k_1 \\ 0 & -lk_1 & 0 & lk_1 \\ lk_1 & 0 & lk_1 & 0 \\ -k_2 & k_2 & -k_2 & k_2 \end{pmatrix} \begin{pmatrix} \delta_f \\ \delta_r \\ \delta_b \\ \delta_l \end{pmatrix} \tag{13}$$

The influence of propeller and blade aerodynamics are complicated and difficult to model, and only becomes significant at high velocities; thus, they are excluded from the model [1]. The model presented by Equations 1 - 4 were used in this study. The Coriolis terms qr, pr, and pq are neglected as the values are small in cases when the rotational movement is slight. The dynamic equations can then be simplified into,

$$\ddot{p}_n = \frac{F}{m}(-\cos\phi \sin\theta \cos\psi - \sin\phi \sin\psi) \tag{14}$$

$$\ddot{p}_e = \frac{F}{m}(-\cos\phi \sin\theta \sin\psi + \sin\phi \cos\psi) \tag{15}$$

$$\ddot{h} = g - \frac{F}{m}(\cos\phi \cos\theta) \tag{16}$$

$$\ddot{\phi} = \frac{1}{I_{xx}}\tau_\phi \tag{17}$$

$$\ddot{\theta} = \frac{1}{I_{yy}}\tau_\theta \tag{18}$$

$$\ddot{\psi} = \frac{1}{I_{zz}}\tau_\psi \tag{19}$$

2.2 Measurement of the second moment of inertia

The moment of inertia of a quadcopter is a major influence in the quadcopter dynamics as seen in Equation 4. There are several methods of calculating the moments of inertia of a quadcopter which include modelling and simulation of bifilar pendulum [6], treating the quadcopter as four small masses surrounding a larger central mass at a distance [4], and combining the measured moments of each

section of the quadcopter [7]. The moment of inertia of the quadcopter used in this study was determined by the spherical central mass and 4-point masses method. This method by which the quadcopters mass can be expressed as four point-masses each of mass *m* surrounding at *l* a spherical central mass of mass *M* and radius *R* as expressed in Fig. 4 simplifies the calculations but requires validation from experimental measurements. The measured masses of the quadcopter, their distances with respect to the *x*, *y*, and *z* axes and the calculated value of the moment of inertia are given in Table 1.

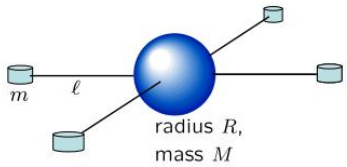

Fig. 4 - The spherical central mass of mass *M* surrounded by 4-point masses of mass m at a distance *l* from the center

Table 1 - Boundary conditions and values

Mass and distances of the quadcopter model	Value	Unit
Central mass, *M*	1.150	kg
Motor mass, *m*	0.245	kg
Radius, *R*	0.054	m
Distance to *x*-axis, l_x	0.213	m
Distance to *y*-axis, l_y	0.213	m
Distance to *z*-axis, l_z	0.302	m
Moment about the *x*-axis, I_{xx}	0.046	kg·m^2
Moment about the *y*-axis, I_{yy}	0.046	kg·m^2
Moment about the *z*-axis, I_{zz}	0.091	kg·m^2

2.3 Controller design

The control algorithm for the quadcopter was developed in LabVIEW for hover and low-speed forward flight conditions. The control algorithm is based on the Proportional-Integral-Derivative (PID) controller which was widely used for closed-loop control application. The general form of the PID controller is given as follows.

$$e(t) = x_d(t) - x(t) \tag{20}$$

$$u(t) = K_p e(t) + K_i \int_0^t e(t)\,dt + K_d \frac{de(t)}{dt} \tag{21}$$

where $u(t)$ is the control input, $e(t)$ is the difference between the desired state $x_d(t)$ and the present state $x(t)$, and K_p, K_i and K_d are the terms of the PID controller [1]. The terms apply corrections to the calculated error value of a system which is the difference between the desired set point of a variable and the measured process variable. The proportional term K_p produces an output that is proportional to the current error value of the system and has a large contribution to the output change. The integral term K_i produces an output that is proportional to the magnitude and duration of the error value and controls the speed by which the control system reduces the error value. The derivative term K_d is based on the rate of change in error value over time and works to change the stability and response of the control system. The traditional PID structure is shown in Fig. 5.

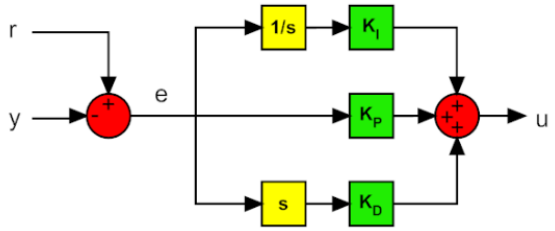

Fig. 5 - Traditional PID structure [8].

Most commercially available quadcopters use PID controllers in their control circuit due to its simple structure and ease of implementation. In this work, the PID cascade control is used. Cascade control is a control method in which two controllers are used with the output of the first controller becoming the setpoint of the second controller. The feedback loop of one controller is nested inside the feedback loop of the other controller [9]. Fig. 6 shows the structure of the PID cascade used in this work.

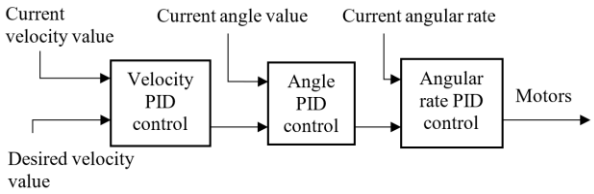

Fig. 6 - The PID cascade control method used

2.4 X-Plane flight simulator

X-Plane is a commercially available flight simulator that can be used to simulate the flight responses of aircraft with a high degree of accuracy. The software uses blade element theory to produce the forces and moments acting on an element of the aircraft surfaces and then applying the processed results to the whole aircraft [11]. In this work, X-Plane 9 is used to provide the quadcopter model and to simulate its dynamics. The flight simulator is connected to the controller in LabVIEW using UDP (user datagram protocol) connection. Fig. 7 shows the data input and output selection interface and the data selected to be sent through UDP to the controller in this work. Commands from the controller to the simulation are sent through the same UDP connection. The numbers on the list of data are the group numbers while the data are sent through individually numbered channels.

UDP connection enables the synchronization of commands, data processing and system responses in the communication between the LabVIEW controller and the X-Plane simulator by using a simple transmission model with no guarantee on data receiving, ordering or integrity. The data packets sent through UDP may go missing, arrive out of order or appear duplicated as the error checking and correction is not done in the protocol, avoiding the

overhead processing, thus increasing the speed the data packet is sent. In real-time systems, dropping data packets is more preferred compared to waiting for delayed data packets. The data packets carry the selected flight parameter data with each parameter receiving an identifying numeric label [12]. In this work, the X-Plane simulator and the LabVIEW controller is implemented on the same computer.

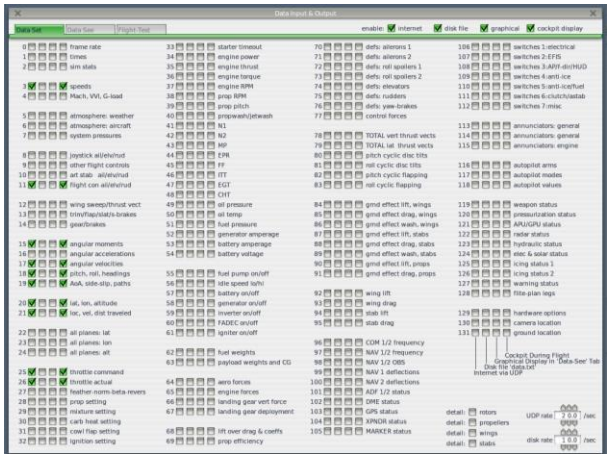

Fig. 7 - X-Plane data input and output selection window

Plane-Maker is an X-Plane program that allows the user to create any type of aerial vehicle. The program uses a graphical interface allowing the creation of an aircraft based on its physical specifications, such as weight, wingspan, engine and wing area. The characteristics of the aircraft modelled in Plane-Maker is used by the X-Plane simulator to predict its flight characteristics [13]. The model details are in Table 2.

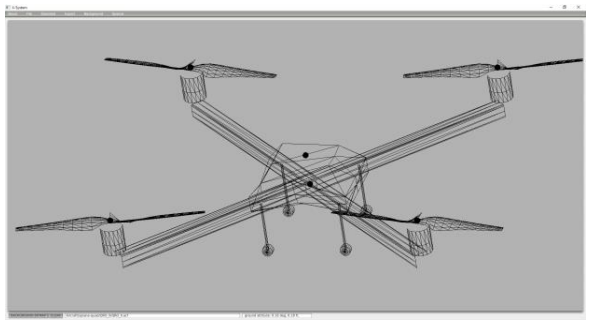

Fig. 8 - Quadcopter model in Plane-Maker

Fig. 9 - Quadcopter model in X-Plane

Table 2 - Model details

Parameter	Value
Weight	1.0 kg / 2.2 lbs
Body radius	0.18 feet
Arm thickness	0.1 feet
Arm angle	43.5°
Battery	1000 watt-hours
Propeller	
Radius	0.42 feet
Root chord	0.8 inches
Tip chord	0.3 inches
Design RPM	4000 rpm
Design	Clark-Y
Engine	
Lateral arm	0.7 feet
Longitudinal arm	0.7 feet
Vertical arm	0.2 feet
Nacelle radius	0.05 feet
Maximum allowable power	0.02 hp

2.5 Flight test

Flight tests were conducted in the simulation to validate the designed control algorithm by assessing the quadcopter's performance in hovering and low-speed flight. The hovering test will be performed by setting the quadcopter to produce thrust to reach a height of 50 feet above the ground and assessing its performance when input in roll, pitch and yaw angles is applied. The low-speed flight will be performed by setting the quadcopter to fly forward at hover height.

3. Results and Discussion

3.1 Correlation between parameters

Fig. 10 and 11 shows the response of the simulated quadcopter to the forward and lateral velocity input of 5 m/s. Orange lines represent the controller input values and yellow lines represent the quadcopter responses. There was no appreciable overshoot in the forward and lateral speed responses of the quadcopter. The rise time for both forward speed and lateral speed is approximately 3 seconds while the settling time for both graphs is approximately 6.5 seconds. The tuned PID gain values enabled the simulated quadcopter to have a fast and stable response to the inputted speed changes. The shape of both graphs shows an overdamped second-order transfer function response, with the time constants for both forward speed and lateral speed about 2.65 seconds. The performance of the velocity controllers is shown in Table 3 and PID controller value in Table 4.

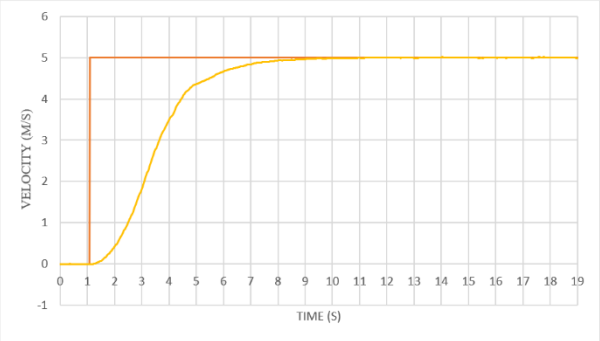

Fig. 10 - Forward velocity response

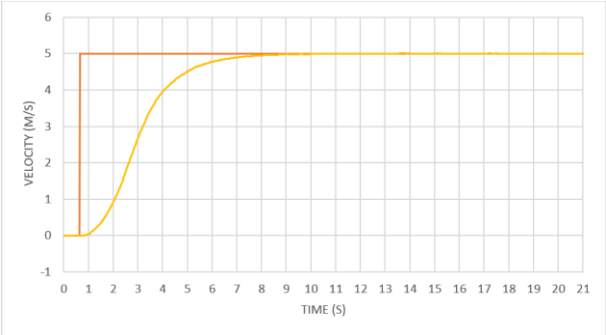

Fig. 11 - Lateral velocity response

Table 3 - Velocity controller performance

Performance parameter	Forward velocity, u	Lateral velocity, v
Rise time (s)	3.40	3.325
Settling time (s)	6.575	6.425
Time constant (s)	2.65	2.65
Steady-state error (m/s)	4.988	4.996

Table 4 - Velocity PID controller values

Controller Gains	Proportional K_p	Integral K_i	Derivative K_d
Forward velocity	-3	1.667	0.005
Lateral velocity	3	1.667	0.005

3.2 Altitude control results

Fig. 12 shows the response of the simulated quadcopter to the altitude input of 50 feet. The orange line represents the controller input values and the yellow line represent the quadcopter responses. There was a small overshoot of about 3.9 percent in the altitude response of the quadcopter. The rise time for altitude response is about 2.8 seconds while the settling time is about 4.1 seconds. The shape of the graph follows the general shape of second-order transfer function responses, with the response having a time constant of 2.4 seconds and a peak time of 3.9 seconds. The altitude controller has a small steady-state error of about 0.001 feet and a response delay of 0.1 seconds. The performance of the altitude controller is shown in Table 5 and the PID controller values in Table 6.

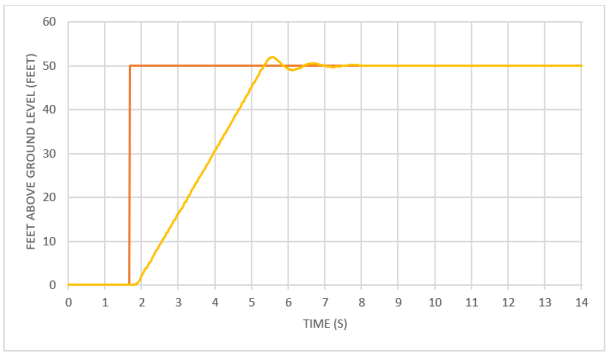

Fig. 12 - Altitude response at 50 ft reference.

Table 5 - Altitude controller performance

Performance parameter	Values
Rise time (s)	2.775
Settling time (s)	4.075
Response delay (s)	0.1
Peak time (s)	5.55
Overshoot (%)	3.931
Steady-state error (ft)	0.001

Table 6 - Altitude controller PID values

Controller Gains	Proportional K_p	Integral K_i	Derivative K_d
Altitude	0.5	0.1	0
Altitude rate	0.8	0	0

3.3 Attitude control results

Fig. 13 to 15 shows the response of the simulated quadcopter to the pitch and roll angle input of 10 degrees without the velocity controller active. Fig. 15 shows the response to a yaw rate input of 1 radian per second. Yellow lines represent the controller input values and grey lines represent the quadcopter attitude responses. Rise time for pitch response is 2.5 seconds, roll response is 2.6 seconds, and the yaw rate response is 0.1 second. The settling time for pitch, roll and yaw rate response is 4.7, 4.8 and 0.7 seconds respectively. The yaw rate response has a large overshoot of 29.8 percent with the roll and pitch response having no overshoot. The shape of the yaw rate response graph follows an undamped second-order transfer function response. The pitch and roll responses fluctuate due to the controller responses to minor disturbance in the attitude values. The performance of the attitude controllers is shown in Table 7 and the PID controller values in Table 8.

Table 7 - Attitude controller performance

Performance parameter	Pitch, θ	Roll, ϕ	Yaw rate, $\dot{\psi}$
Rise time (s)	2.475	2.625	0.1
Settling time (s)	5.0	4.825	0.7
Peak time (s)	-	-	0.23
Overshoot (%)	-	-	29.807
Steady-state error	-	-	-

Table 8 - Attitude control PID values

Angular variable	Proportional K_p	Integral K_i	Derivative K_d
Pitch, θ	0.8	0	0
Pitch rate, $\dot{\theta}$	0.4	0.1	0
Roll, ϕ	1.2	0	0
Roll rate, $\dot{\phi}$	0.6	0.1	0.1
Yaw rate, $\dot{\psi}$	0.45	0	0

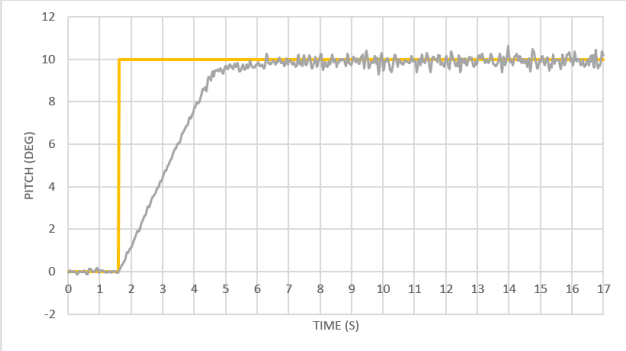

Fig. 13 - The quadcopter UAV's pitch response at 10° reference

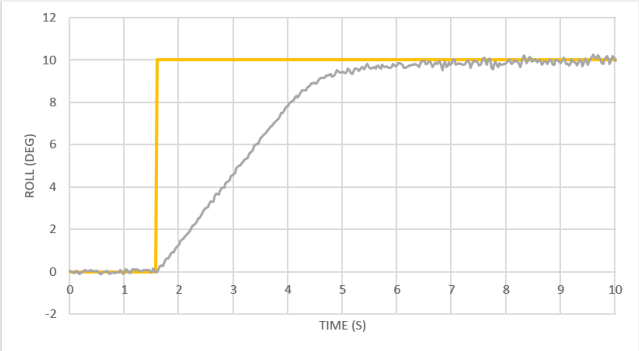

Fig. 14 - The quadcopter UAV's roll response at 10° reference

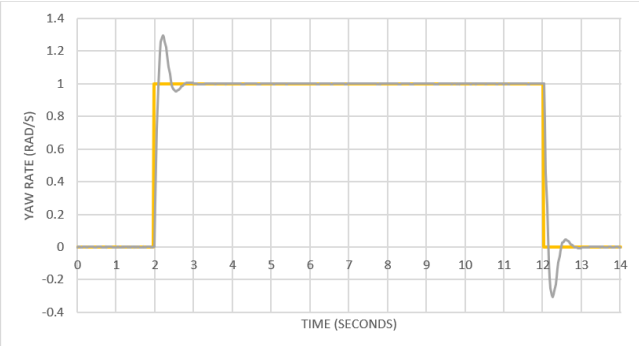

Fig. 15 - The quadcopter UAV's yaw rate response due to 1 rad/s reference

4. Conclusion

The objectives of this work were the development of a dynamic model and control simulation of a quadcopter in order to develop a control algorithm for quadcopter hover and low-speed flight conditions and the validation of the controller performance. The model of a quadcopter was implemented in the flight simulator software X-Plane and the controller was implemented in LabVIEW. The controller was connected to the simulation containing the model using a UDP connection. The PID values of the controllers were then tuned manually to reach specific performance. The performance of the manually tuned controller was observed and recorded. The controllers implemented in this work require further fine-tuning of the gain values to decrease the rise and settling times and reduce the overshoot. It is recommended that in future work to use the PID autotune VIs that are available in LabView to obtain more accurate PID gain values. Another recommendation is to use different control methods and design the controller to achieve positional tracking flight. It is also recommended to implement the designed controller on real quadcopter hardware after performing the comparison between the different controllers in a simulated environment.

Acknowledgement

The authors thank to the all industries partner and Universiti Tun Hussein Onn Malaysia (UTHM) for equipment and to the Aeronautical Engineering Department of UTHM for their support.

Appendix

The PID controller code implemented.

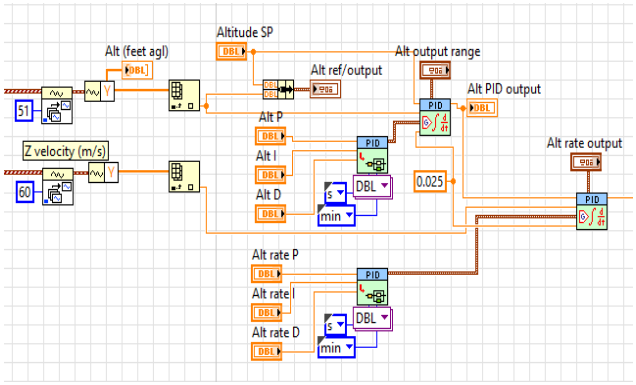

Figure A-1: Altitude PID control

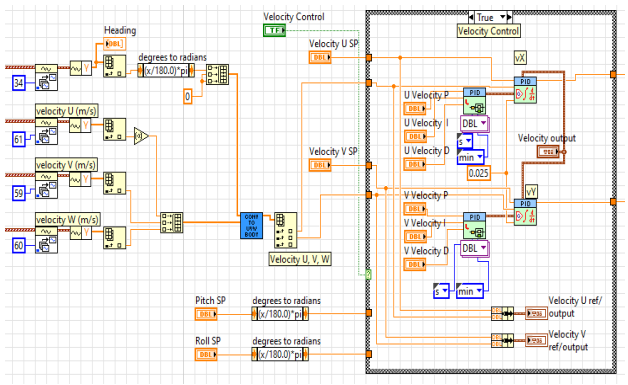

Figure A-2: Forward and lateral velocity PID control

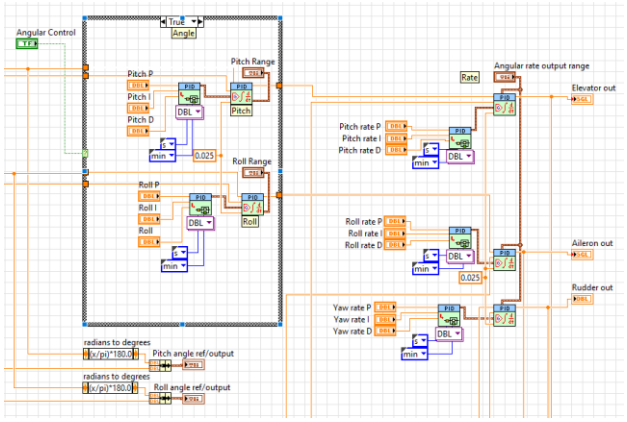

Figure A-3: PID control for roll and pitch angles (left) and angular rates (right).

The yaw rate controller is implemented instead of a yaw controller because in X-Plane, the yaw values are between 0° and 360°. A yaw controller would cause the quadcopter to spin when the yaw angle is set to 0° as when the quadcopter yaw angle value goes below 0°, the yaw angle value in X-Plane would change to 359°. This causes the controller to spin the quadcopter to reach the yaw angle value set.

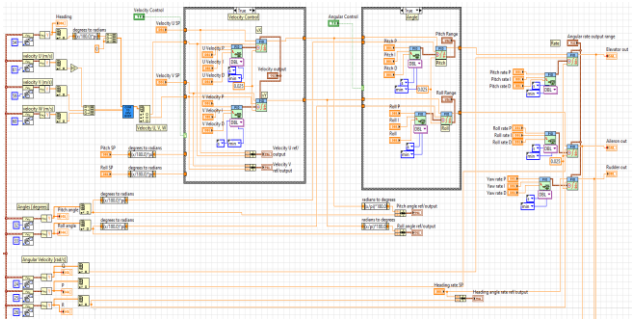

Figure A-4: PID control for forward and lateral velocity (center left), pitch and roll angles (center right), and pitch roll and yaw rates (left).

References

[1] T. Luukkonen, "Modeling and control of a quadcopter." Aalto University, Espoo, 2011.

[2] "The World's Smallest Quadcopter - Hammacher Schlemmer."Available:https://www.hammacher.com/product/worlds-smallest-quadcopter. [Accessed: 27-May-2019]

[3] X. Zhang, X. Li, K. Wang, and Y. Lu, "A Survey of Modelling and Identification of Quadrotor Robot," *Abstr. Appl. Anal.*, 2014 (2014): 1–16

[4] R. Beard, "Quadrotor Dynamics and Control Rev 0.1," *All Faculty Publications*. 2008

[5] M. F. Silva *et al.*, "Design of angular PID controllers for quadcopters built with low cost equipment," in *2016 20th International Conference on System Theory, Control and Computing*, (2016): 216–221

[6] M. R. Jardin and E. R. Mueller, "Optimized Measurements of Unmanned-Air-Vehicle Mass Moment of Inertia with a Bifilar Pendulum," *J. Aircr.*, 46 (3) (2009): 763–775

[7] T. Jiinec, "Stabilization and Control of Unmanned Quadcopter," Czech Technical University, 2011.

[8] T. Bresciani, "Modelling, Identification and Control of a Quadrotor Helicopter," Lund University, 2008.

[9] W. Bolton, "CHAPTER 13 Control Systems," in *Instrumentation and Control Systems*, (2015): 281–302.

[10] D. F. de Castro and D. A. dos Santos, "A software-in-the-loop simulation scheme for position formation flight of multicopters," *J. Aerosp. Technol. Manag.*, 8(4) (2016): 431–440

[11] "How X-Plane Works | X-Plane." [Online]. Available: https://www.x-plane.com/desktop/how-x-plane-works/. [Accessed: 12-Dec-2019]

[12] A. Kaviyarasu, P. Sivaprakash, and K. Senthilkumar, "Lecture Notes in Electrical Engineering: Preface," *Lect. Notes Electr. Eng.*, 188 (2013): 343–355

[13] A. Bittar, N. M. F. De Oliveira, and H. V. De Figueiredo, "Hardware-in-the-loop simulation with X-plane of attitude control of a SUAV exploring atmospheric conditions," *J. Intell. Robot. Syst. Theory Appl.*, 73 (1-4), (2014): 271–287

[14] H. V. Figueiredo and O. Saotome, "Simulation platform for quadricopter: Using Matlab/Simulink and X-plane," *Proc. - 2012 Brazilian Robot. Symp. Lat. Am. Robot. Symp. SBR-LARS 2012*, pp. 51–55, 2012.

[15] H. Talla M. ElKholy, "Dynamic Modeling and Control of a Quadrotor Using Linear and Nonlinear Approaches," The American University in Cairo (AUC), 2014