UNIVERSITY OF
TORONTO

AER1216

Fundamentals of Unmanned Aerial Vehicles

Simulation Analysis of Fixed-Wing and Multi-Rotor Aircraft

Project Report

Fall 2023

Group 8

Student	Student Number
Derek Mau	1003487817
Syed Shoaib Hasan	1004306985
Wanshun Xu	1006473004

Table of Contents

List of Figures

List of Tables

OVERVIEW

Background: This project covers the development of simulation models for fixed wing and multirotor UAV configurations. An unmanned aerial vehicle (UAV) is a part of a bigger unmanned aircraft system (UAS), which can also include a ground control station (CGS), a control link (like a radio), and more [1]. They can also be set up to be flown autonomously, using autopilot systems and different controllers that account for dynamic conditions.

Objectives: The objective of the fixed wing development is to create an autopilot system for the Aerosonde UAV, which can receive navigational commands of airspeed, heading, and altitude, and compute the control surface and throttle commands. Additional calculations for range, endurance, and accounting for wind conditions will also be tested. For the multirotor, the equivalent range and endurance parameters will be calculated. Also, a linear simulation model will be made for hovering at a point and tracing a flat horizontal circle.

Theoretical Framework: The simulation framework for the fixed wing was partially given by the TA's, which includes a visual rendering of the aircraft (draw_aircraft), and the aircraft dynamics block, whose derivative parameters were filled in using a non-linearized dynamics model given by the reference textbook [2]. The simulation framework for the multirotor was also mostly given, with changes to state space variables for the dynamic modelling, and parameter changes in the path controller for either hovering or tracing a circle.

Methods and procedures: For the fixed wing, closed feedback control loops were made for the altitude hold, heading control, and speed control systems, input into the aircraft dynamics model, and plotted. Testing maneuvers included steady level flight, a coordinated turn, and a 100m climb. The control loops were tuned with PID controllers to ensure stable response times, and desired control values were changed to test the robustness of the controller, such as varying wind conditions, a higher climb, different initial altitude, etc. For the multirotor, changes in the path controller were made for either hovering or tracing a circle. A reference and actual trajectory were both plotted in the simulation to show the discrepancies when using a linearized dynamics model. While the control loops in the path controller were already set up with reasonable proportional and derivative gains, small changes were made to illustrate what impact they have on the trajectory error.

Significance: This project signifies the importance of controller design and simulation testing for UAVs. These tests can be used to build a more robust control system, automate a specific flight path, or be expanded to account for additional interferences, such as loss of reception (the UAV "going home", or reverting to its original location), loss of thrust on one motor (for fixed wing, gliding to a specific flight path, for multirotor, initiating an emergency landing protocol), etc. Testing for these additional scenarios in simulation ensures safe flight operation during emergencies and allows us to better understand how control systems work.

1. FIXED-WING UAS DEVELOPMENT

1.1 Flight Range and Endurance

The range of an aircraft is defined as the total distance, measured with respect to the ground, an aircraft can fly on one tank of fuel. It's based on fuel consumption, and therefore dependent on engine performance. For a propellor-based aircraft, the specific fuel consumption (SFC, or c_p), is the key metric, defined as the weight of fuel consumed per unit hour per unit power. For us, it's given as 0.6651 g/hr/W. Now we need the formula for maximum range of an aircraft, which occurs when a minimum amount of fuel is consumed per unit distance [3].

$$R_{max} = \frac{\eta}{c_p}\frac{C_L}{C_D} ln(\frac{W_0}{W_1}) = \frac{\eta/c_p}{2\sqrt{KC_{D_0}}} ln(\frac{W_0}{W_1})$$

Where η is the efficiency of the propellor, W_0 is the weight of the aircraft with a full tank of fuel, W_1 is the weight of the aircraft with no fuel, C_{Do} is the zero lift drag coefficient, and $K = \frac{1}{\pi e(AR)}$, where e is the Oswald's efficiency factor, and AR is the aspect ratio. Note that this is for a steady level flight at a set altitude. Similarly, the endurance of an aircraft is the total time it stays in the air on one tank of fuel. For steady level flight, the formulation for endurance is similar to the range formulation [3]:

$$E_{max} = \frac{\eta}{c_p}\sqrt{2\rho S}\left(\frac{C_L^{3/2}}{C_D}\right)_{max}\left(\frac{1}{\sqrt{W_1}} - \frac{1}{\sqrt{W_0}}\right)$$

Where ρ is the density of the air at a specified altitude, and S is the wing area. Also, $(C_L^{3/2}/C_D)_{max} = 1/(4\ C_{Do})*(3\ C_{Do}/K)^{3/4}$. Before calculating these values, we must know what η is. To get this, we need the cruise velocity of the aircraft, which can be found to be V = 13.877m/s. Because we have C_D, we can also find drag, and therefore thrust at steady state. Next, we need the motor speed n (rev/s). Given that we have C_T versus J graph, we can pick a random speed n, go to the corresponding J value, and corresponding C_T value, solve for thrust, and compare with the steady value. After a bit of trial and error, we find n = 31.5 rev/s, J = 0.1843, C_T = 0.221, C_P = 0.0486, and $\eta = C_TJ/C_P \approx 0.8$. Now, we can calculate the above formulas, after adjusting c_p value:

If c_p = 0.6651 g/hr/W, then c_p *(9.81m/s^2) * (1kg/1000g) * (1hr/3600s) = 1.81239E-6 (N/Ws).

$R_{max} = \frac{\frac{\eta}{cp}}{2\sqrt{K(CDo)}} \ln\left(\frac{W0}{W1}\right)$ = (0.8/1.81239E-6(1/m))/2*$\sqrt{0.0232 \cdot 0.03}$(ln(13.1/9.1) = 3,047,950.89m or 3047.95 km.

$E_{max} = \frac{\eta}{cp}\sqrt{2\rho S}(C_L^{3/2}/C_D)_{max}\ln\left(\frac{1}{\sqrt{W1}} - \frac{1}{\sqrt{W0}}\right)$ = 0.8/1.81239E-6(N/Ws) *

$\sqrt{2 \cdot \frac{1.225kg}{m^3} \cdot 0.55m^2}\left(\left(\ln\left(\frac{1}{\sqrt{9.1kg}} - \frac{1}{\sqrt{13.1kg}}\right)\right)\left(\frac{1}{4 \cdot 0.03}\right)\left(\left(3 \cdot \frac{0.03}{0.0232}\right)\right)^{3/4}\right.$ = 651,484.62s = 180.96 hr.

1.2 Dynamics Model Development

Development of the fixed-wing dynamics model was split into two major sections: kinematics and dynamics, and forces and moments. The main reference used to aid in the design of these sections was the suggested textbook "Small Unmanned Aircraft, Theory and Practice" by Randal W. Beard

and Timothy W. McLain. This textbook will also serve as the main source of knowledge for all proceeding sections of fixed-wing aircraft development. By solving the equations needed for the dynamics model, our group was then able to translate these equations to code which was then implemented in the functions provided by the course instructor.

Kinematics and Dynamics

This section focuses on deriving the calculations necessary to determine the forces on the six axes of freedom in both the inertial and vehicle reference frames. A figure showing the 6 axes of motion can be found in appendix A-1.

The inertial velocity vectors are denoted as $\dot{P}_n$, $\dot{P}_e$, and $\dot{P}_d$ and represent the change in position of the north, east and down positions respectively. The vehicle reference vectors are denoted by u_r, u_r, and u_r and represent the airspeed of the vehicle. As for the angular velocity vectors, inertial vectors are defined as $\dot{\phi}$, $\dot{\theta}$, and $\dot{\psi}$ while the vehicle vectors are p, q, and r. The equations needed to calculate these values can be referenced in appendix A-1.

Forces and Moments

Calculating the forces and moments of the aircraft will lead to developing equations that will take inputs from the control surfaces δ_e, δ_a, δ_r and δ_t. In this section, we mainly calculate forces and moments generated from three sources, gravity, aerodynamics and propulsion. Equations used to model these forces can be found in appendix A-1.

1.3 Altitude/Speed/Heading Controls Development (MATLAB/Simulink)

When designing the control for the altitude, speed and heading for the UAS, chapter 6 of the textbook was used as a basis for design. Additionally, input timings and commands were then added to the system to properly execute the maneuvers listed in part 5 for the fixed wing project description. The development of the Simulink model can be divided into 3 sections: heading, speed and altitude controls. Much of the Simulink model development is attributed to chapter 6 of the textbook [2]. A detailed view of the Simulink model can be found in appendix A-2.

Vehicle Controls

The development of the vehicle controls is built upon the basics of feedback loops. Each value that is passed through the system is compared back to a reference value where an error is calculated through the PID control block and further processed to the next control variable.

For the speed control section, a reference value of 19.4 m/s (70 km/h) was given as a parameter of the project and is set to be the target speed the system much as keep. This signal is then processed to a delta thrust variable where it was found through the simulation that a constant value of 0.28 must be held for the simulation to fly at this value. Speed is compared to Va which is the scalar value of the wind speed vector components described in section 1.2.

Heading control took an input of ψ and compared this value with ϕ which later compared to p and then finally inputted as aileron controls to the aircraft dynamics model. ψ was decided to be the main control variable because this represented the true heading of the aircraft, which would be essential to maneuver 2. Control diagrams from the textbook laid out how the control variable

errors would flow into aileron control. This diagram was adopted and implemented into our simulation.

Similar to heading control, Altitude hold control followed as similar principle of identifying the main variable needed for the system and then deriving the next variables needed to control the main selected variable. In this case, it was determined that altitude was the essential variable which would then flow into θ and then elevator controls. From simulation testing, it was demonstrated that steady state flight required an elevator hold angle of -11.35 degrees (negative since the elevators push "down" to force the aircraft up) and was thus used as an initial hold input for the aircraft. This initial value assisted with the quick response to altitude hold.

PID Tuning

For the Simulink model, the use of PID controls was used to tune the error of each variable using predefined criteria to produce the best response for the system. Initially, the use of the Ziegler–Nichols turning method was used but after some testing, it was deemed to not effectively tune the system to the predetermined criteria set. This may be due to several factors, but the results given from this tuning method did not result in a steady state value for the PID controller. Thus, it was decided that for the sake of convenience that the Simulink internal automatic tuner would be used to progress the state of the project.

When using the automatic tuner provided by Simulink, the main 2 criteria used were the settling time and overshoot percentages. For a quick but stable response, a settling time of 20 seconds was used in addition to limiting the overshoot of the response to a maximum of 25%. These two design parameters allowed the system to respond effectively to the inputs provided by the maneuver controls. To reach the set parameters, trial and error was used while adjusting the response time and PID robustness to result in the satisfaction of the design criteria.

Maneuver Controls

To ensure that each maneuver was executed in successive order, the use of timing, limit and comparison blocks were implemented in the Simulink model to control these actions. For maneuver 1, altitude hold at 1000m for 250m, the system simply commands 1000m altitude to the altitude control system. To determine the 250m travel distance, a delay block was used with a set delay length of 72 clock ticks. The delay length was determined by knowing the system needed approximately 13 seconds traveling at 19.44 m/s with each second being roughly 6 clock ticks. Maneuver 2 was triggered once 250m was traveling. Using the heading rate formula provided in the textbook, integrating the heading change rate resulted in a steady change in heading until the desired value for 90 degrees was reached. Maneuver 3 used a similar strategy to maneuver 2 by detecting when the 90-degree heading was reached to trigger an altitude change. The altitude command is like maneuver 1. The control system has limitations such as being unable to descend in altitude due to how the climb rate was interpreted in the control system (a negative value needed for descending action) and being unable to turn in a counterclockwise direction (similar issue to maneuver 3)

1.4 Simulation Results

The results of the simulation are logged in the plots located in appendix A-1. Overall, the simulation performed as expected according to the design of the Simulink model. There were no major discrepancies that were unexplainable, and all the logged movements of the aircraft were well within the expected limits. The following are some observations made during the 3 maneuvers outlined by the project requirements.

Table 1: Maneuver 1 - Hold Altitude 1000m for 250m

Observation	Analytic Quantity
Time to Stabilization:	18.798 s
Time to 250 m:	12.873 s

It was observed that initially the altitude would fluctuate due to what is assumed to be the elevator control surface reaching a steady state hold angle. This resulted in some time needed for the 1000m to be reached and held for the appropriate distance. After the altitude was reached, the time needed to hit the 250 m checkpoint was extremely close to the theoretical value of 12.857 seconds.

Table 2: Maneuver 2 - Coordinated 90 Degree Turn with 250 m Radius.

Time to Execute:	21.079 s
Time to Stabilization:	45.693 s
Radius Error:	-6.8 % (17 m)
Heading Error:	4% (3.61 deg)

The only significant observation from maneuver 2 was that some rudder input was needed to stabilize the aircraft while performing the coordinated turn. This input can be seen in the result plots located in the appendix, but it was an interesting note that sideslip control would be needed while the aircraft banked and turned. Normally it was assumed that only aileron inputs would be needed to perform this action.

Table 3: Maneuver 3 - Climb to 1100m Altitude Within 30s

Time to Execute:	31.758 s
Time to Stabilization:	61.088 s
Altitude Error:	0.19% (1102 m)

Finally for maneuver 3, the climb was calculated using a desired end time of 30 seconds with an altitude gain of 100m. Initially the altitude would overshoot by only 2m but due to the slow system response and low overshoot, the simulation was able to perform this maneuver perfectly. Some major observations from the plots indicated oscillations in the theta, elevator, w and q values. This oscillation can be explained by the simulation having to increase the pitch angle and settling down to the steady state value necessary for the climb. Once the angle was achieved, the system was able to settle until the target altitude was reached. Once achieved, a similar oscillation can be seen with an identical explanation.

1.5 Theoretical Expectations vs Results

- Cruise flight angle of attack: Assuming steady state (L=W), we can find CL, as L = W = (½)$\rho S V^2$ C_L = (13.1kg * 9.81m/s^2) = (0.5)(1.112 kg/m^3)(0.55 m^2)(19.44m/s)^2 (C_L), or C_L = 1.11 = C_{Lo} + $C_{L\alpha}$ $\boldsymbol{\alpha}$, where $\boldsymbol{\alpha}$ = 0.241 rad, assuming $\boldsymbol{\delta_e}$ = 0 . However if $\boldsymbol{\delta_e} \neq 0$, then C_L = 1.11 = C_{Lo} + $C_{L\alpha}$ $\boldsymbol{\alpha}$ + $C_{L\,\delta e}$ $\boldsymbol{\delta_e}$. In a coordinated turn, $\boldsymbol{\delta_e}$ = -11.27, so $\boldsymbol{\alpha}$ = 0.0741 rad.
- Yaw rate: In a coordinated turn, the yaw rate $\dot{\Psi} = \frac{V cos\gamma}{R}$, and assuming the flight path angle γ = 0, the relation is just V/R, or 19.44m/s / 250m = 0.0776 (1/s)
- Climb rate: $\dot{h} = V\,sin\gamma$, where we can find the γ = tan^{-1}(3.33m/s / 19.44m/s) = 9.72° , where the 3.33m/s rate of change in altitude (100m in 30s), and V = 19.44m/s. Therefore $\dot{h}$ = 3.28m/s

Table 4: Theoretical vs Simulated values

Theoretical vs Simulated Values			
	Theoretical	Simulated	Error (%)
Cruise flight angle α (rad)	0.241	0.2052	-14.9
Yaw rate Ψ (rad/s)	0.0776	0.0843	8.6
Climb rate (R/C) (m/s)	3.28	2.956	9.87

1.6 Wind Effects

To incorporate wind into the simulation, chapters 2.3 and 4.4 of the reference textbook [2] were used to further develop the Simulink model. One core assumption that was made for wind effects was to omit the turbulence generated by gust as it was determined that gust effects would generate a unique and substantially different environment to analyze the aircraft. It was further assumed that once gust effects were detected by the UAV, manual operation of the vehicle was necessary for continued safe operation. For the implementation of wind, the following formula was used:

$$\mathbf{V}_w^b = \begin{pmatrix} u_w \\ v_w \\ w_w \end{pmatrix} = \mathcal{R}_v^b(\phi, \theta, \psi) \begin{pmatrix} w_{n_s} \\ w_{e_s} \\ w_{d_s} \end{pmatrix} + \begin{pmatrix} u_{w_g} \\ v_{w_g} \\ w_{w_g} \end{pmatrix}$$

This implementation resulted in 3 inputs, one for the north, east and down winds respectively. To determine safe operation of the vehicle under wind conditions, the main testing parameter was simply when the system could no longer recover from the disturbance during the execution of the predetermined maneuvers. Our group defined the “out of control” scenario as when any rotational axis was no longer able to stabilize after 30 seconds of simulation run time. An additional factor was that each direction of wind was tested independently as testing the wind effects together would be too complicated for the scope of this project. As a result, the following values were determined to be the limits of safe operation of this UAV.

Table 5: Windspeed limits

Wind Direction	**Speed (m/s)**
North	7
East	3
Down	12

2. MULTI-ROTOR DRONE DEVELOPMENT

2.1 Quadrotor Performance Calculation

Based on the information given the total weight of the quadrotor drone is $m = 420g$; $C_D = 0.97$; Reference area $S = 0.01m^2$; This quadrotor has 4 APC 8x6 Slow Flyer propellers. The battery is a 3 Cell 1500 mAh battery. Standard earth gravity $g = 9.81\frac{m}{s^2}$; Normal density of air $\rho = 1.293\frac{kg}{m^3}$.

Calculation:

The Diameter of propeller can be calculated $D = 0.2032m$. To solve this problem, Forward Flight-Momentum Theory will be used. At a given Speed,V we can then solve for α_D,v ,T ,P_{ind},P_{tot} as follows:

1. Find frame drag $D_f = \frac{1}{2}\rho S_{ref} C_{Df} V^2$ at $\alpha_D = 0$. Assume α_D dose not effect the drag. Otherwise need to iterate to find a solution.
2. Solve $\alpha_D = \tan^{-1}\left(\frac{D_f}{W}\right)$
3. Square both sides of Eqn 1 replace T^2 by $W^2 + D_f^2$ and re-arrange to get,

$$v^4 + (2Vsin\alpha_D)v^3 + V^2v^2 - \frac{\left(W^2 + D_f^2\right)}{(2\rho A)^2} = 0$$

4. Where A is total area of all discs, and position real root of equation gives v , then

$$P_{ind} = Tv \text{ and } P_{tot} = T(v + Vsin\alpha_D)$$

The calculation process outlined above will be executed utilizing Matlab. As the specific speed of the quadrotor was not provided, the team made the assumption of speeds ranging from 0 to 20 m/s, with increments set at 0.5 m/s.

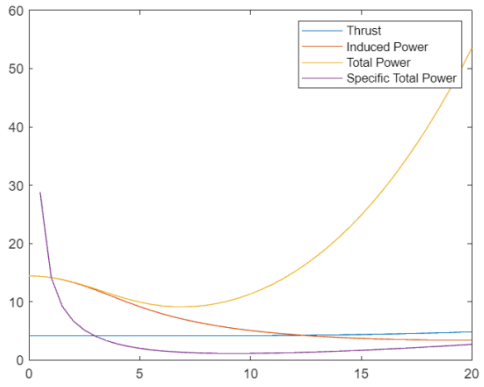


Figure 1: Plot for Forward Flight

The figure presented above depicts the Quadrotor plot based on momentum theory for forward flight, encompassing speeds ranging from 0 to 20 m/s with 0.5 m/s increments. The plot includes data for Thrust, induced Power, Total Power, and Specific Total Power. For the determination of

maximum endurance and maximum range, our focus lies solely on Total Power. The objective is to identify the minimum Total Power required for the quadrotor's forward flight.

$$\min(P_{total}) = 9.1029$$

$$V_{\min(P_{total})} = 15$$

Team can use the endurance formula: $t_e = \frac{E_b \eta_m \eta_e}{\min(P_{total})}$ to find the maximum endurance for quadrotor and alos the Range formula: $t_e = \frac{\frac{E_b \eta_m \eta_e}{P_{prop}}}{Speed_{V_{\min(P_{total})}}}$. where the $\eta_m = 0.7; \eta_e = 0.8$.

After Matlab Calculation team find the final answer:

$$Max(Range) = 25812m \rightarrow 25.812km$$

$$Max(Endurance) = 3687.4s \rightarrow 1h2\,\min$$

2.2 Matlab/Simulink (linear) Simulation Model:

The team experimented with various approaches to construct the Quadrotor Simulink model. Ultimately, the team found valuable guidance on modeling from Qian and incorporated those insights into the final simulation.

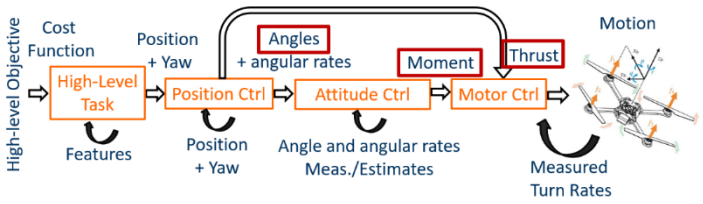


Figure 2: The cascade/nested Control Architecture [4]

The team has chosen a cascade/nested Control Architecture for the quadrotor drone, involving translational and attitude control components. Translational control, using current and desired positions, employs position PD control to achieve the desired lift. Attitude control calculates the necessary angle change for the quadrotor based on the desired lift, and this angle is then used to determine the required Torque. The control system, illustrated in Figure 3 of the appendix, simplifies the Simulink model by using the provided quadrotor state space information. The state-space function directly converts the desired angle into the quadrotor's state, modifying the original attitude control to include an angle conversion block. This block transforms the desired lift into the desired angle, serving as a conversion rather than a control block. Additionally, dynamic modeling utilizes the state-space function to transform desired angles (pitch, yaw, roll) into position (x, y, h) and speed (u, v, vz). An initial position control block is introduced to modify the quadrotor's initial position, incorporating an "if" function to apply an additional constant value when x, y, and h are set to 0. This allows for a flexible starting point beyond the default (0,0,0) coordinates.

2.3 Task Simulation

a. Hover at 2 meters above original

Appendix Figure outlines the position control block developed incrementally for point-to-point tasks, employing a Proportional-Derivative (PD) controller for rapid response and improved tuning. This strategy ensures the drone's precise trajectory control, dynamically adjusting speed for efficient maneuvers.

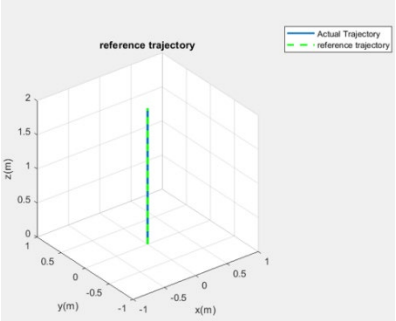


Figure 3: Plot for Hover at 2 meters above original

The accompanying flight path figure depicts the drone's adherence to the reference trajectory (red line), achieving a precise stop at (0,0,2). Despite the control system's simplicity, minimal PD parameter tuning was required. The PD controller's effectiveness is evident in the drone's accurate trajectory tracking and its ability to reach the specified stopping point with minimal deviation.

b. Following a Horizontal Circle

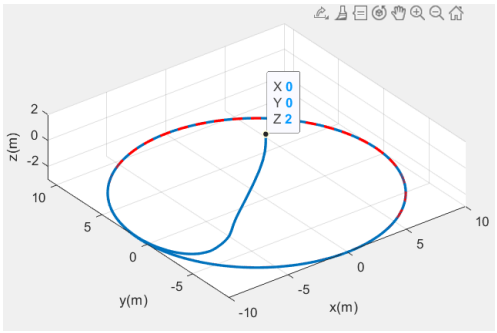


Figure 4: Plot for Following a Horizontal Circle (Start point at 0,0,2)

Following instructions, the drone takes off from (0,0,2), navigating a trajectory centered around (x = 0, y = 1, z = −3) with R = 10m and v = 1m/s. Figure 8 visually portrays the quadrotor's reference (red line) and actual (blue line) trajectories. The group's designed quadrotor adeptly follows this path, demonstrated by images from different starting points. Notably, the algorithm leverages angular velocity to influence the quadrotor's motion, introducing an arc before entering the circular trajectory. This strategic use enhances the trajectory-following capabilities for smooth navigation along the designated circular path.

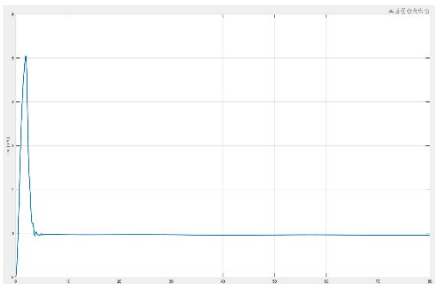


Figure 5: Plot for Normal of Speed over time

The Simulink simulation results plotted by the team demonstrate the successful completion of the assigned task by the drone. Upon careful observation of the speed plot, it becomes apparent that the Proportional-Derivative (PD) controller used by the team exhibits noticeable overshooting. This characteristic could potentially be refined and enhanced through further tuning of the controller parameters.

An insightful analysis of the second-order system characteristics reveals favorable traits in the UAV's performance. Specifically, the settling time is short, and there is a lack of pronounced oscillations. These characteristics are indicative of a well-designed control system. The short settling time suggests that the drone quickly reaches a stable state after perturbations, and the absence of significant oscillations points to a controlled and stable response. Overall, these findings signify a commendable performance of the drone in meeting the specified control objectives.

c. Test the Controller

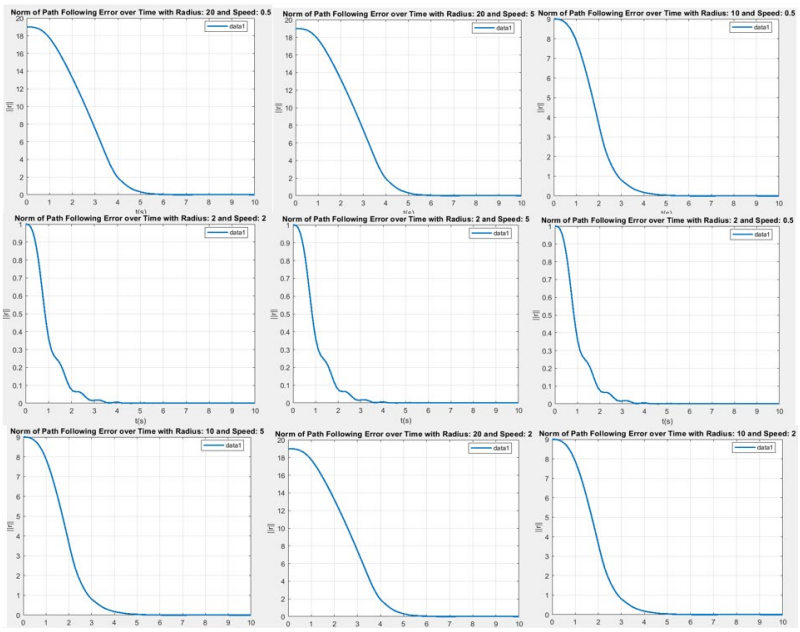


Figure 6: Plot for normal of error at different radius and speed

Figure 8 serves as a valuable tool for analyzing the relationship between ||r|| (magnitude of the error vector) and v (cruising speed) and ||r|| and R (radius of the circular trajectory). Observations reveal that increasing R while keeping v constant prolongs the UAV's stability attainment, aligning with expectations due to the larger trajectory radius requiring more time for circular path traversal. Conversely, when R is too small, the drone deviates, causing irregularities in ||r||, emphasizing the need for a well-chosen trajectory radius. Interestingly, with constant R, variations in v minimally impact the UAV, affirming the PD controller's effective pre-entry speed control. This underscores the controller's role in maintaining trajectory accuracy.

2.4 PD gains and Performance

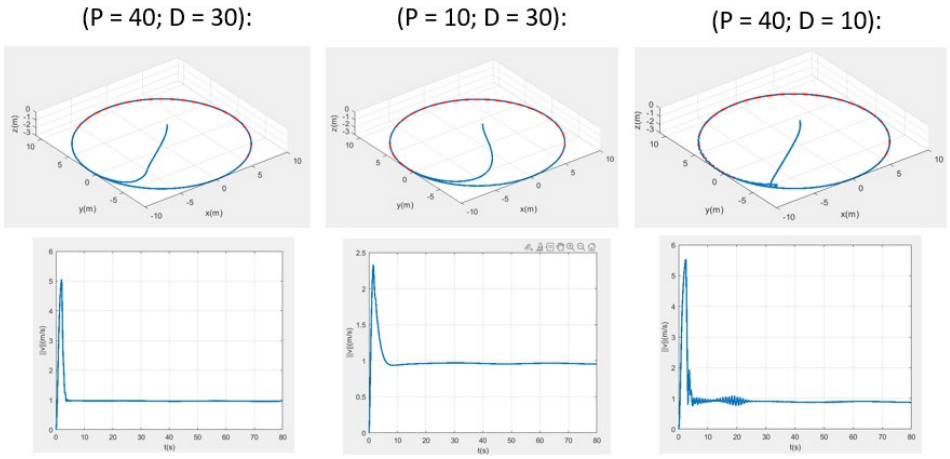


Figure 7: Plot for Circle trajectory with different PD values

By tuning the Proportional (P) and Derivative (D) parameters of the PD controller, the team observed that reducing the P value minimized overshooting but prolonged settling time, while a smaller D value increased overshoot, oscillations, and stabilization time; trajectory analysis revealed that a decreased P value resulted in a larger curve before entering the circular path, whereas a reduced D value caused the drone to enter the trajectory nearly straight but with increased oscillations and a longer stabilization time.

3. CONCLUSIONS

3.1 Fixed-Wing UAS Development

Outside of being more comfortable with using Simulink, the team has learned a fair bit about closed feedback control loops when developing an autopilot system. While a good chunk of the code to plot and simulate the fixed wing was given, learning the non-linearized dynamic parameters gave us a deeper understanding of the relationship between control surfaces and aircraft physics. This was apparent when seeing discrepancies in theoretical and actual simulation results, despite how small they may seem. Additionally, a lot was learned about tuning the system with PID controllers, including its limitations. These included adding a reference equilibrium value to the throttle and aircraft surface commands, as well as saturation limits to prevent the aircraft from dangerous maneuvers like stalling. If given more time, additional maneuvers could also be simulated with control over the aircraft surfaces, such as crosswind landings, or stall recovery. In fact, with proper simulation modelling of the engine and propeller, simulated range and endurance can also be found and compared to the theoretical values calculated.

3.2 Multi-Rotor Development

While a good chunk of the linear simulation model was also given for the quadrotor, the team learned a lot about the linearized dynamic model through the translational and attitude control components. The use of a PD controller proved useful in simulation tasks, and even when analyzing for different circular path trajectories, the only notable trajectory deviations were observed during a large change in radius. While tuning the PD controller wasn't necessary, additional observations about the P and D gain values showed the former's effect on minimizing overshoot and settling time, and the latter's effect on oscillation and stabilization time. If given more time, additional maneuvers could also be tested, such as path tracing a figure 8, or interrupting a maneuver at a certain time for an emergency landing.

3.3 Roles And Responsibility Breakdown

Team Member	Main Responsibilities
Syed Shoaib Hasan	Overview, Fixed wing range and endurance, conclusion. Also helped with making Simulink model for the multirotor.
Wanshun Xu	Multirotor range and endurance calculation, build multirotor Simulink model with Shoaib, Matlab function for analysis the multirotor performance
Derek Mau	Fixed-Wing: Dynamics model (coding and calculations), Simulink model (altitude, speed, heading, wing, maneuver timing), Fixed wing analysis, Team coordinator, Minutes keeper, Presentation and report formatter.

References

[1] "Unmanned Aircraft Systems / Drones," 2015. [Online]. Available: https://rmas.fad.harvard.edu/unmanned-aircraft-systems-drones.

[2] T. W. M. Randal W. Beard, Small Unmanned Aircraft, Theory and Practice, Princeton: Princeton University Press, 2012.

[3] J. F. M. III, "Chapter 6. range and endurance," [Online]. Available: https://pressbooks.lib.vt.edu/aerodynamics/chapter/chapter-6-range-and-endurance/.

[4] Longhao Qian, "Multirotor_dynamics_2_longhao-2," 30 11 2023. [Online]. Available: https://q.utoronto.ca/courses/314592. [Accessed 18 12 2023].

Appendix

A-1 Fixed-Wing Dynamic Equations

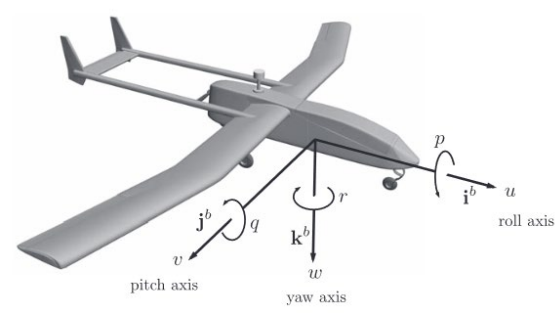


Figure 8: 6 Axes of Motion on Standard Aircraft

$$\begin{pmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{p}_d \end{pmatrix} = \begin{pmatrix} C_\theta C_\psi & S_\phi S_\theta C_\psi - C_\phi S_\psi & C_\phi S_\theta C_\psi + S_\phi S_\psi \\ C_\theta S_\psi & S_\phi S_\theta S_\psi + C_\phi C_\psi & C_\phi S_\theta S_\psi - S_\phi C_\psi \\ -S_\theta & S_\phi C_\theta & C_\phi C_\theta \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix}$$

$$\begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix} = \begin{pmatrix} rv - qw \\ pw - ru \\ qu - pv \end{pmatrix} + \frac{1}{m} \begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix},$$

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & \sin\phi \tan\theta & \cos\phi \tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix}$$

$$\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \begin{pmatrix} \Gamma_1 pq - \Gamma_2 qr \\ \Gamma_5 pr - \Gamma_6 (p^2 - r^2) \\ \Gamma_7 pq - \Gamma_1 qr \end{pmatrix} + \begin{pmatrix} \Gamma_3 l + \Gamma_4 n \\ \frac{1}{J_y} m \\ \Gamma_4 l + \Gamma_8 n \end{pmatrix}$$

$$\begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix} = \begin{pmatrix} -mg \sin\theta \\ mg \cos\theta \sin\phi \\ mg \cos\theta \cos\phi \end{pmatrix}$$

$$+ \frac{1}{2}\rho V_a^2 S \begin{pmatrix} C_X(\alpha) + C_{X_q}(\alpha)\frac{c}{2V_a} q + C_{X_{\delta_e}}(\alpha)\delta_e \\ C_{Y_0} + C_{Y_\beta}\beta + C_{Y_p}\frac{b}{2V_a} p + C_{Y_r}\frac{b}{2V_a} r + C_{Y_{\delta_a}}\delta_a + C_{Y_{\delta_r}}\delta_r \\ C_Z(\alpha) + C_{Z_q}(\alpha)\frac{c}{2V_a} q + C_{Z_{\delta_e}}(\alpha)\delta_e \end{pmatrix}$$

$$+ \frac{1}{2}\rho S_{\text{prop}} C_{\text{prop}} \begin{pmatrix} (k_{\text{motor}}\delta_t)^2 - V_a^2 \\ 0 \\ 0 \end{pmatrix}, \qquad (4.18)$$

$$\begin{pmatrix} l \\ m \\ n \end{pmatrix} = \frac{1}{2}\rho V_a^2 S \begin{pmatrix} b\left[C_{l_0} + C_{l_\beta}\beta + C_{l_p}\frac{b}{2V_a}p + C_{l_r}\frac{b}{2V_a}r + C_{l_{\delta_a}}\delta_a + C_{l_{\delta_r}}\delta_r\right] \\ c\left[C_{m_0} + C_{m_\alpha}\alpha + C_{m_q}\frac{c}{2V_a}q + C_{m_{\delta_e}}\delta_e\right] \\ b\left[C_{n_0} + C_{n_\beta}\beta + C_{n_p}\frac{b}{2V_a}p + C_{n_r}\frac{b}{2V_a}r + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r\right] \end{pmatrix} + \begin{pmatrix} -k_{T_p}(k_\Omega \delta_t)^2 \\ 0 \\ 0 \end{pmatrix}. \quad (4.20)$$

A-2 Fixed-Wing Code

```
    % Air Data
        rho = 1.225;
        mass_empty = 9.1;
        mass_fuel = 4;
        mass = mass_empty+mass_fuel;
        g = P.g;

        %windless condition

        ur = u-uw;
        vr = v-vw;
        wr = w-ww;

        Va = sqrt(ur^2 + vr^2 + wr^2);
        alpha = atan(wr/ur);
        beta = asin(vr/Va);

    % rotation matrix
        R_bv = [cos(theta)*cos(psi)                          cos(theta)*sin(psi)                          -sin(theta);
                sin(phi)*sin(theta)*cos(psi)-cos(phi)*sin(psi)  sin(phi)*sin(theta)*sin(psi)+cos(phi)*cos(psi)  sin(phi)*cos(theta);
                cos(phi)*sin(theta)*cos(psi)+sin(phi)*sin(psi)  cos(phi)*sin(theta)*sin(psi)-sin(phi)*cos(psi)  cos(phi)*cos(theta)];

        R_vb = R_bv.';
```

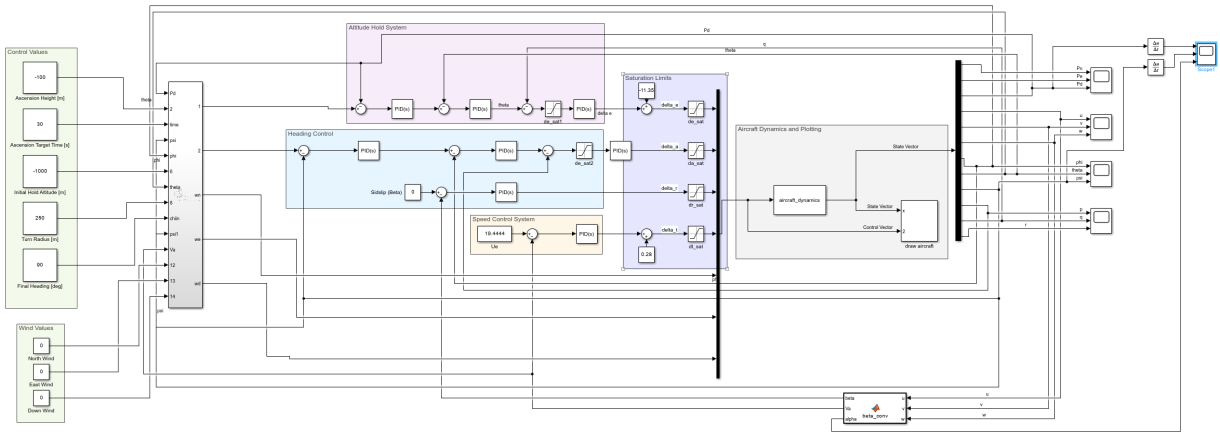
```
    % Aerodynamic Coefficients
    % compute the nondimensional aerodynamic coefficients here

        %Longitudinal
        CLo = 0.28;
        CDo = 0.03;
        Cmo = -0.02338;

        CLalpha = 3.45;
        CDalpha = 0.30;
        Cmalpha = -0.38;

        CLq = 0;
        CDq = 0;
        Cmq = -3.6;

        CLdeltae = -0.36;
        CDdeltae = 0;
        Cmdeltae = -0.5;

        %Lateral
        CYo = 0;
        Clo = 0;
        Cno = 0;

        CYbeta = -0.98;
        Clbeta = -0.12;
        Cnbeta = 0.25;

        CYp = 0;
        Clp = -0.26;
        Cnp = 0.022;

        CYr = 0;
        Clr = 0.14;
        Cnr = -0.35;

        CYdeltaa = 0;
        Cldeltaa = 0.08;
        Cndeltaa = 0.06;
```

```
        CYdeltar = -0.17;
        Cldeltar = 0.105;
        Cndeltar = -0.032;

    % aerodynamic forces and moments
    % compute the aerodynamic forces and moments here
        b = 2.8956;
        S = 0.55;
        e = 0.9;
        c = 0.18994;
        sub1 = b/(2*Va);
        sub2 = c/(2*Va);
        AR = b^2/S;

        %Linear Model used
        CL_alpha = CLo + CLalpha*alpha;%G
        CD_alpha = CDo + CDalpha*alpha;%G
        Cm_alpha = Cmo + Cmalpha*alpha;%G

        %Longitudinal
        aero_fx = (1/2)*rho*Va^2*S...
                    *(-CD_alpha*cos(alpha) +    CL_alpha*sin(alpha))...
                    +(-CDq*cos(alpha) +         CLq*sin(alpha))*sub2*q...
                    +(-CDdeltae*cos(alpha) +    CLdeltae*sin(alpha))*delta_e;%G

        aero_fz = (1/2)*rho*Va^2*S...
                    *(-CD_alpha*sin(alpha) -    CL_alpha*cos(alpha))...
                    +(-CDq*sin(alpha) -         CLq*cos(alpha))*sub2*q...
                    +(-CDdeltae*sin(alpha) -    CLdeltae*cos(alpha))*delta_e;%G

        aero_m =  (1/2)*rho*Va^2*S*c*(Cmo + Cmalpha*alpha + Cmq*sub2*q + Cmdeltae*delta_e);%G

        %Lateral Forces
        aero_fy = (1/2)*rho*Va^2*S*  (CYo + CYbeta*beta + CYp*sub1*p + CYr*sub1*r + CYdeltaa*delta_a + CYdeltar*delta_r);%G
        aero_l =  (1/2)*rho*Va^2*S*b*(Clo + Clbeta*beta + Clp*sub1*p + Clr*sub1*r + Cldeltaa*delta_a + Cldeltar*delta_r);%G
        aero_n =  (1/2)*rho*Va^2*S*b*(Cno + Cnbeta*beta + Cnp*sub1*p + Cnr*sub1*r + Cndeltaa*delta_a + Cndeltar*delta_r);%G

        aero_forces =  [aero_fx;aero_fy;aero_fz];
        aero_moments = [aero_l;aero_m;aero_n];

    % propulsion forces and moments
    % compute the propulsion forces and moments here

    Sprop = 0.2027;
    Cprop = 1;
    kmot = 80;

    fprop = (1/2)*rho*Sprop*Cprop*[(kmot*delta_t)^2-Va^2;0;0];

    % gravity
    % compute the gravitational forces here

    fg = R_bv*[0;0;mass*g];

    % total forces and moments (body frame)

    tot_forces = fg + aero_forces + fprop;

    fx=tot_forces(1,1);
    fy=tot_forces(2,1);
    fz=tot_forces(3,1);

    tot_moments = aero_moments;

    l = tot_moments(1,1);
    m = tot_moments(2,1);
    n = tot_moments(3,1);
```

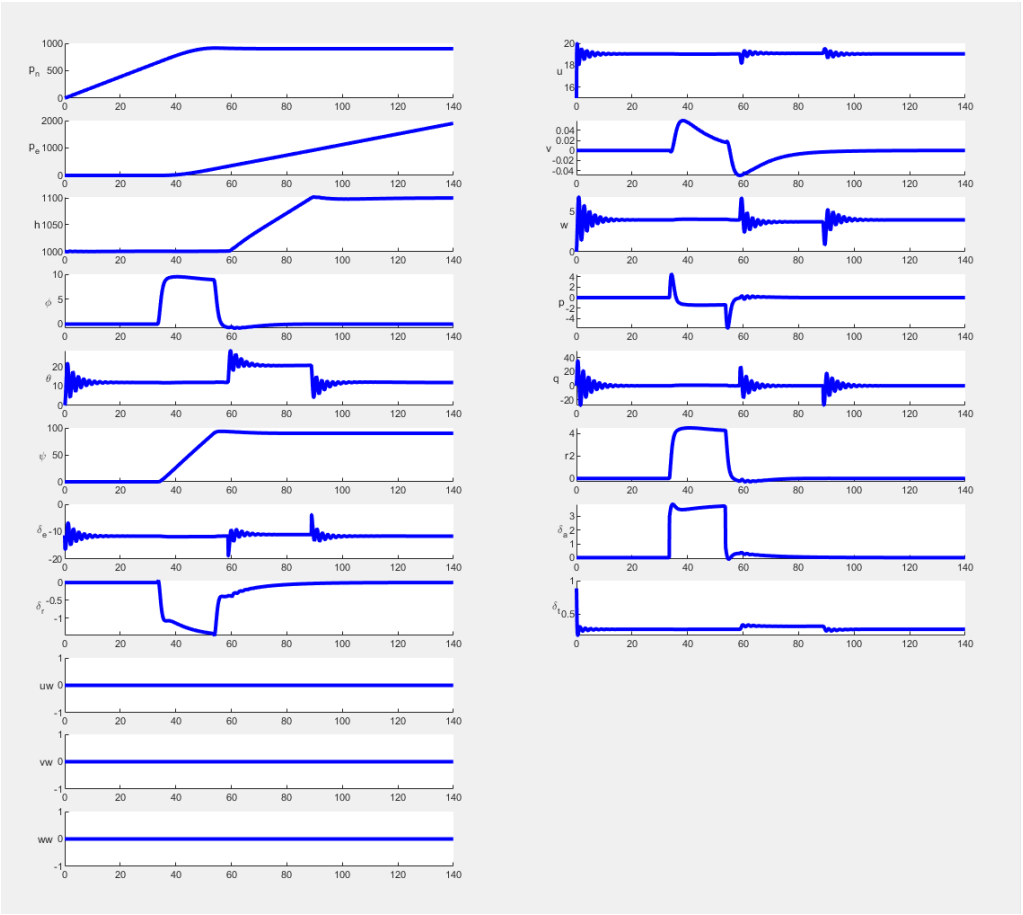
d

```
431       % state derivatives
432       % the full aircraft dynamics model is computed here
433
434       P_mat = R_vb*[ur;vr;wr]; %G
435
436       pndot = P_mat(1,1); %G
437       pedot = P_mat(2,1); %G
438       pddot = P_mat(3,1); %G
439
440       V_mat = [r*vr - q*wr;
441                p*wr - r*ur;
442                q*ur - p*vr;]+...
443                (1/mass)*[fx;fy;fz]; %G
444
445       udot = V_mat(1,1); %G
446       vdot = V_mat(2,1); %G
447       wdot = V_mat(3,1); %G
448
449       eulerangs = [1 sin(phi)*tan(theta) cos(phi)*tan(theta);
450                    0 cos(phi) -sin(phi);
451                    0 sin(phi)*sec(theta) cos(phi)*sec(theta)]...
452                   *[p;q;r]; %G
453
454       phidot = eulerangs(1,1); %G
455       thetadot = eulerangs(2,1); %G
456       psidot = eulerangs(3,1); %G
457
458       pdot = k1*p*q - k2*q*r + k3*l + k4*n; %G
459       qdot = k5*p*r - k6*(p^2 - r^2) + m/P.Iy; %G
460       rdot = k7*p*q - k1*q*r + k4*l + k8*n; %G
```

A-3 Fixed-Wing Simulink Model



A-4 Fixed-Wing Simulation Plots



B-1 Quadcopter Performance Matlab Calculation Code

```
clear all
clc
% Given Parameters
g = 9.81; % m/s^2
Cd = 0.97;
S = 0.01;
W = 0.42*g; % in N
N_Cell = 3;
mAh = 1500;
roh = 1.293; % Density of air
D = 0.2032; % Diameter in m

%0th order model
Eta_m = 0.7; % motor is 70% efficient
Eta_e = 0.8;  % ESC is 80% efficient
E_b = N_Cell*3.7*(mAh/1000)*3600;

% Calculation
A = 4*pi*(D^2)/4; % There are 4 blade
syms v
Speed = 0:0.5:20;
P_tot = zeros(length(Speed),1);
T = zeros(2,1);
P_ind = zeros(2,1);
p_tot = zeros(2,1);
for i = 1:length(Speed)
    V = Speed(i);
    Df = (roh*S*Cd*(V^2)/2);
    Alfa_D = atan(Df/W);
    Eq = (v^4)+(2*V*sin(Alfa_D))*(v^3)+(V^2)*(v^2)-(W^2+Df^2)/((2*roh*A)^2) == 0;
    Induce_v = double(max(solve(Eq,v, 'Real',true)));
    T(i) = sqrt(W^2+Df^2);
    P_ind(i) = sqrt(W^2 + Df^2)*(Induce_v);
    P_tot(i) = sqrt(W^2+Df^2)*(Induce_v+V*sin(Alfa_D));
end
specific_power = P_tot./Speed';
figure
plot(Speed,T);
hold on;
plot(Speed,P_ind);
plot(Speed,P_tot);
plot(Speed,specific_power);

legend({'Thrust','Induced Power','Total Power','Specific Total Power'})
hold off
```

```
[P_tot_min,P_tot_V] = min(P_tot)
```

```
P_tot_min = 9.1029
P_tot_V = 15
```

```
x=Speed(P_tot_V);
Max_Range = (E_b*Eta_e*Eta_m)/(P_tot_min/x)
```

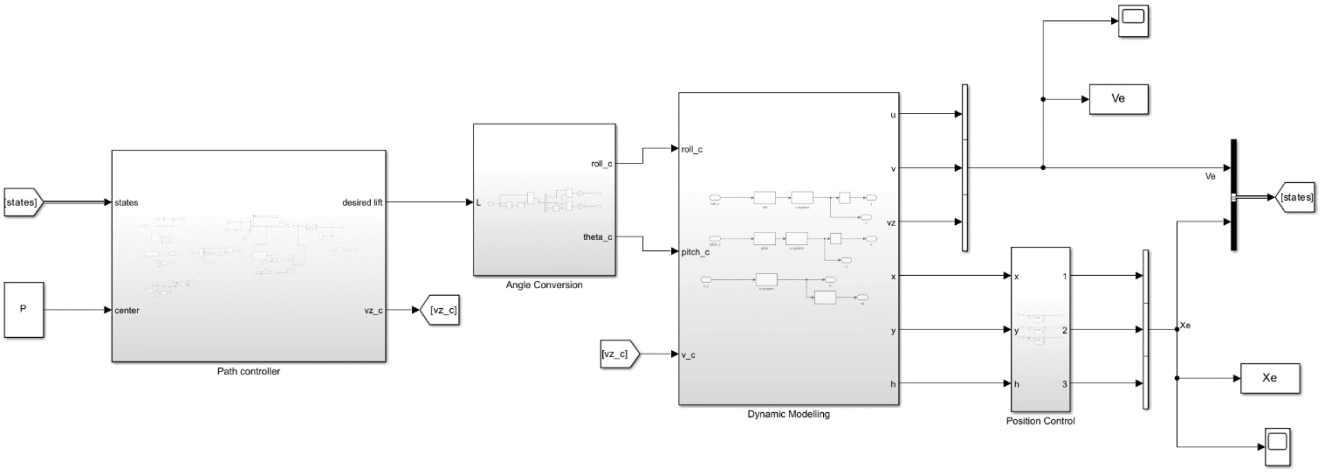
```
Max_Range = 2.5812e+04
```

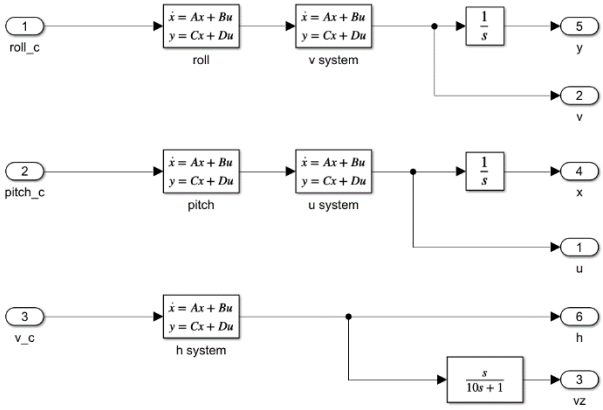
```
Max_Endurance = (E_b*Eta_e*Eta_m)/(P_tot_min)
```

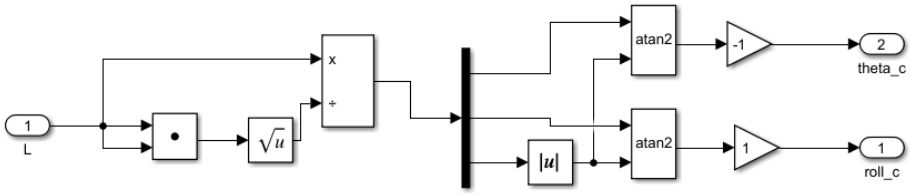
```
Max_Endurance = 3.6874e+03
```

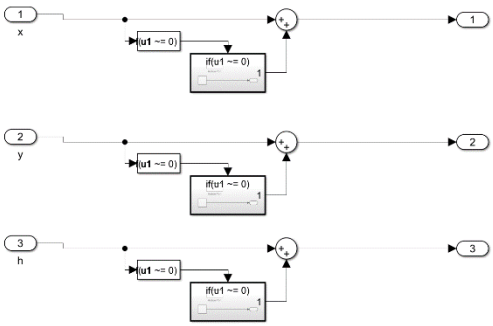
g

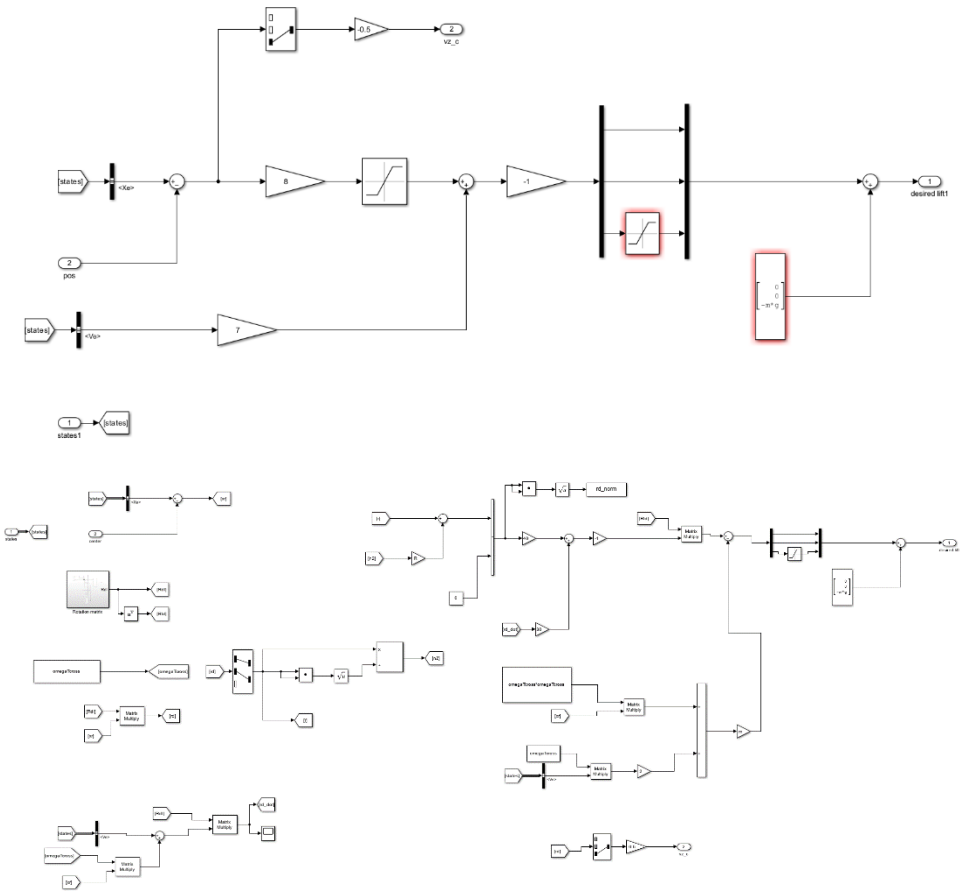
B-2 Quadcopter Simulink Model











B-3 Quadcopter Matlab Call-function with Plot

```
clc, clearvars, close all
m = 0.42;
g = 9.81;
P = [0;0;2];
sim('position_stabilization',10)
Xe = ans.Xe;
figure(1)
% actual trajectory
plot3(Xe.data(:, 1), Xe.data(:, 2), Xe.data(:, 3),'LineWidth', 2)
grid on
axis equal
xlabel('x(m)')
ylabel('y(m)')
zlabel('z(m)')
title('Quadrotor actual trajectory');
hold on

%reference trajectory
Start = [0;0;0];
plot3([Start(1), P(1)], [Start(2), P(2)], [Start(3), P(3)], '--g', 'LineWidth', 2);
grid on
axis equal
xlabel('x(m)')
ylabel('y(m)')
zlabel('z(m)')
title('reference trajectory');
legend('Actual Trajectory','reference trajectory')
```

```
clc, clearvars, close all
m = 0.42;
g = 9.81;
vd = 1;
R= 10;
omegaT = vd/R;
P = [0;1;-3];
W = [0;0;0.1];
omegaTcross = util_crossmatrix(W);
sim('Multirotor',80)

figure(1)
% actual trajectory
plot3(Xe.data(:, 1), Xe.data(:, 2), Xe.data(:, 3), 'LineWidth', 2)
hold on
% reference trajectory
theta=-pi:0.01:pi;
x=P(1)+R*cos(theta);
y=P(2)+R*sin(theta);
z = P(3) * ones(1, numel(theta));
plot3(x,y,z,'--r', 'LineWidth', 2)

grid on
axis equal
xlabel('x(m)')
ylabel('y(m)')
zlabel('z(m)')

figure(2)

Vabs = sqrt( Ve.data(:, 1).^2  + Ve.data(:, 2).^2 + Ve.data(:, 3).^2);

plot(Ve.Time, Vabs, 'LineWidth', 2)
grid on
xlabel('t(s)')
ylabel('||v||(m/s)')

function px = util_crossmatrix(p)
    px = [0 -p(3) p(2);p(3) 0 -p(1);-p(2) p(1) 0];
end
```

```
clc, clearvars, close all
% constant variables
m = 0.42;
g = 9.81;
W = [0;0;0.1];
P = [0;1;-3];
omegaTcross = util_crossmatrix(W);
% controller by varing the radius and speed
speed = [0.5;2;5];
radius = [10;2;20];

for i = 1:length(speed)
    for j = 1:length(radius)
        R = radius(j);
        vd = speed(i);
        omegaT = vd/R;
        sim('Multirotor',10)
        R_normal_plot(rd_norm,R,vd)
    end
end

function R_normal_plot(rd_norm,R,vd)
% Extract data from timeseries
figure;
plot(rd_norm.Time, rd_norm.Data, 'LineWidth', 2)
title(['Norm of Path Following Error over Time with Radius: ',num2str(R),' and Speed: ',num2str(vd)]);
xlabel('t(s)');
ylabel('||r||');
legend('Location', 'Best');
grid on;
end

function px = util_crossmatrix(p)
    px = [0 -p(3) p(2);p(3) 0 -p(1);-p(2) p(1) 0];
end
```

j