M.Eng Project: Magnus Ultralight

Exploration of Initial Control System Architecture for an E-VTOL utilizing Magnus Effect

Eric Xue[1]

[1]Cornell University, Masters of Engineering Student

May 17, 2024

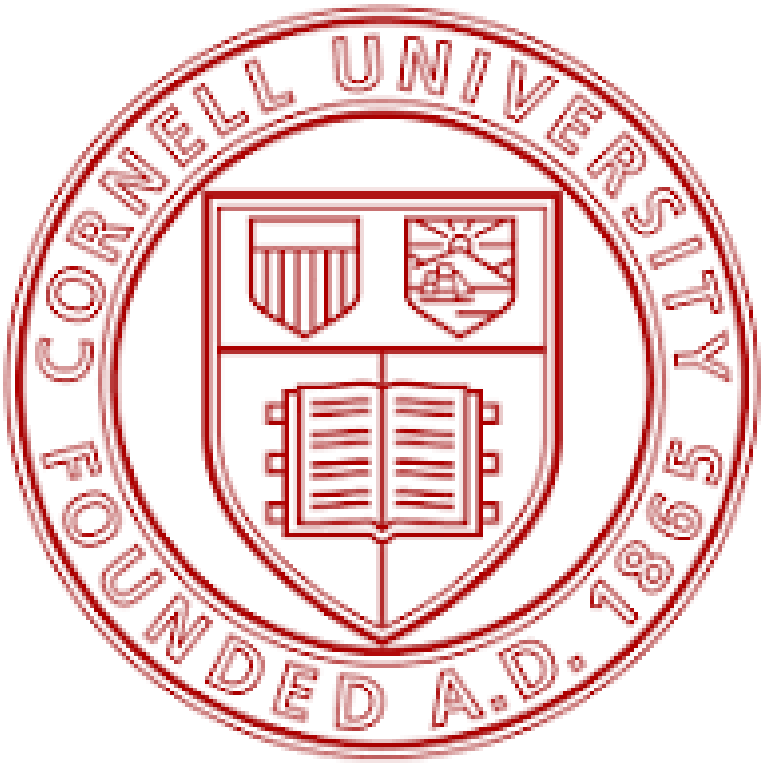


Table of Content

1. Abstract

Magnus Ultralight is a quadcopter with a central spinning object aimed to generate extra lift via the Magnus effect, the interaction between an oncoming airflow and a spinning object which creates a pressure differential. It is theorized that utilizing this to generate enough lift to keep the aircraft afloat, would allow for the quadcopter to tilt forward 90 degrees, obtaining a tiltrotor configuration and using its propeller as propulsion. This allows for the aircraft to achieve greater efficiency while also having higher airspeeds than possible with just a quadcopter formation.

Having a spinning object introduces interesting gyroscopic considerations due to the extra angular momentum. This makes Magnus more susceptible to instabilities not seen from a regular quadcopter due to the coupling of the roll and yaw axis from this gyroscopic torque. Thus, in order for a proper control system architecture to be introduced, a fully non-linear dynamic model must be produced which accounts for this extra term.

After a sufficient model was simulated, the controller was designed. A cascade control scheme was quickly settled on due to the way a quadcopter obtained 6 DOF control via coupling the X and Y motion with pitch and roll. A significant amount of time was spent exploring the controller type, such as LQR and PD control with quaternion and rotational rates. It was seen that linear techniques generally worked well with the inner loop containing the attitude, but due to the non-linearities of the outer loop, the system quickly destabilized once using the same techniques. Ultimately, 6 PIDs were chosen as controllers for the 6 DOFs.Simulations demonstrated that the cascade control scheme with PIDs was successful in achieving the desired system response to step inputs.

2. Introduction

VTOL aircraft, capable of vertical takeoff and landing without runways, are extremely promising for autonomous flight applications. Magnus Ultralight aims to harness the VTOL capabilities of a quadcopter, merging them with a widely observed fluid mechanics phenomenon, the Magnus Effect, to generate lift. This effect happens when a spinning object encounters an oncoming fluid flow, creating a pressure differential. We see this effect every day in sports when the pitcher throws a curve ball or when a soccer player adds a spin to their kick to curve it around a goalie. Magnus Ultralight takes advantage of this by involving a spinning cylinder attached to the airframe, which will accelerate airflow atop the cylinder to create a pressure disparity and generate additional lift. The final goal sees the craft transitioning to a tilt-rotor configuration by pitching forward 90 degrees. This shift allows Magnus to tap into further propulsion from the quadcopter motors, enabling faster cruise speeds compared to conventional quadcopter formations while potentially providing greater efficiency.

The report will dive into the dynamic model of Magnus and outline the preliminary architecture of its control system. Prior work on this neglected the gyroscopic torque seen by having a spinning object producing additional angular momentum, which is non-negotiable once the cylinder is spun up to speeds enough to produce significant lift. After the non-linear dynamical equations are fully derived to include this extra angular momentum, the first goal of a control system is to stabilize Magnus in its quadcopter formation. This will focus on

accommodating the gyroscopic torques while holding altitude to ensure the quadcopter is able to combat the effects once spun up.

Many quadcopters implement a variation of the cascade control scheme presented here, with those that do not employ more complex nonlinear controllers. In order to make this work easily able to be built off of just the fundamentals of linear control theories, linear techniques were explored, ultimately relying on PIDs. This report will explore the theory and work that went into a successful simulation of this control scheme as well as any other potential suggestions for future members building off this work.

3. Kinematics / Dynamics

3.1 Angle Representation

The attitude for a preliminary control design was represented by Euler Angles due to the representation being the most intuitive, which was essential when first tuning the gains for the PIDs. Since the goal of the quadcopter is to reach a roll angle of 90 degrees, the system will face gimbal lock issues without any gimbal locking prevention measures, which makes it tempting to use a quaternion representation. However, since it is still unclear whether the prototype can achieve this configuration, it will be assumed that Euler angles will be sufficient for now until further testing indicates a valid benefit that can be gained from switching over to quaternions. Additionally, using quaternions directly as part of the PID controller typically requires the use of a special nonlinear controller that is designed with this in mind, which is something that can be looked into [1].

Some further notes on this, using quaternions and converting them into Euler angles will also result in gimbal locking issues. However, a potential way around this to include quaternions is to involve the angular rates but this will be limited to a linear controller [2].

With the angle representation selected, a corresponding rotation representation/direction cosine matrix (DCM) must also be chosen. While most aircraft by convention use intrinsic axes for rotation, this report used a set of extrinsic axes due to the simpler visualization. Both representations are fine and mathematically equivalent in describing the orientation of a 3-D object and can be interchanged if a preference exists. The DCM featured is a 3-2-1 Euler Angle Convention around the extrinsic X-Y-Z axis, making the Roll Angle (ϕ) the rotation about an inertial X axis, the Pitch Angle (θ) the rotation about an inertial Y axis, and the Yaw Angle (ψ) being the rotation about an inertial Z axis. Thus, the DCM that describes the orientation of Magnus is as follows:

$$
{}^{\mathfrak{B}}R^{\mathfrak{N}} = \begin{bmatrix} c(\theta)c(\psi) & c(\theta)s(\psi) & -s(\theta) \\ s(\phi)s(\theta)c(\psi) - c(\phi)s(\psi) & s(\phi)s(\theta)s(\psi) + c(\phi)c(\psi) & s(\phi)c(\theta) \\ c(\phi)s(\theta)c(\psi) + s(\phi)s(\psi) & c(\phi)s(\theta)s(\psi) - s(\phi)c(\psi) & c(\phi)c(\theta) \end{bmatrix}
$$

Figure 1: DCM

3.2 Force / Torques

Despite being an underactuated system, a quadcopter can get around this via the property of differential flatness. A system is described as such when there exists a set of outputs such that all the states and inputs can be expressed as a function of that output and its time derivatives[3]. While this property isn't necessary for the control of just the 3 Euler angle and altitude/height, it becomes necessary to map the forces/torques of the 4 motors to the full 6DOF, and how quadcopters are capable of path planning and trajectory control. With this established, a simplified model of the thrust and yaw torque imparted by the four motors is described as follows:

$$\text{Thrust of Motor: } T_{m_i} = k_1 \Omega_i^2 \qquad \text{Reaction Torque of Motor: } \tau_{\psi_i} = (\pm) b \Omega_i^2$$

Figure 2: Force and Torque of Brushless Motors

Several assumptions and simplifications that went into this are missing in multiple papers that employ the same equations, but a detailed walk-through is provided by Gibiansky[4]. This representation of the torques and thrust in terms of the motor speeds is extremely important when considering the ingenuity behind motor placement in the design of a quadcopter. To obtain independent control of all 6 DOF, the motors of a quadcopter are spun such that adjacent motors spin in opposite directions, while diagonal motors spin in the same direction. A sample configuration and the one that will be used in this report is as follows:

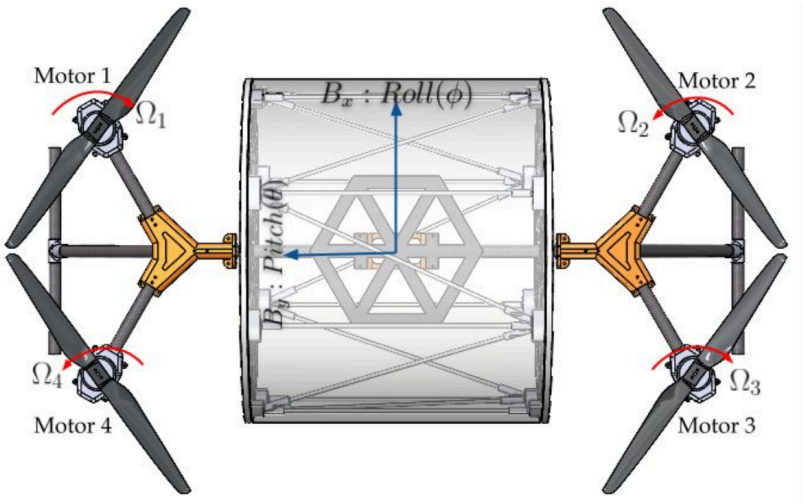


Figure 3: Quadcopter Motor Configuration

From this configuration, it becomes evident that spinning the motors in pairs will lead to independent control over the 3 Euler angles and altitude. By spinning up the motor pairs 2 and 3 or 1 and 4, the thrust imbalance causes a net positive or negative torque respectively about the Roll Axis, inducing a change in the roll angle. By spinning up the motor pairs 1 and 2, or Motors 3 and 4, there is a net negative or positive torque about the Pitch axis. By spinning up the motor pairs 2 and 4 or 1 and 3, there is either a net positive or negative torque about the yaw axis due to

the reaction torque of the motors back onto the aircraft. Lastly, to gain or lose altitude, all 4 four motors need to either spin up or down simultaneously to reduce the overall thrust without causing a torque on the body. This leaves the remaining two DOFs which involve the X and Y translation. While traditional quadcopters are not able to directly control these DOFs, the system is able to take advantage of the independent control of pitch and roll to do so indirectly. By simply pitching in the direction of X or rolling in the direction of Y, the quadcopter can alter the direction of the thrust vector and impart a force in the desired direction, thus providing control over the translational motion. The results of this are summarized below:

$$
\begin{aligned}
F_T &= k_1(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\
\tau_\phi &= Lk_1(\Omega_1^2 - \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \\
\tau_\theta &= Lk_1(-\Omega_1^2 - \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\
\tau_\psi &= b(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2)
\end{aligned}
$$

Figure 4: Force and Torque in terms of Motor Speed.

3.3 Rotational Dynamics / Euler's Torque Equation

The rotational dynamics of the Magnus can be described by Euler's Torque equation with a slight modification to the angular momentum of the overall system. Normally, the angular momentum in the derivations of the torque equation features just the angular momentum of the body itself, but with a spinning object attached to the body with its own angular momentum, the overall angular momentum must be modified. From there, the rest of the derivation remains the same, taking into account the additional time derivative of the angular momentum of the spinning object via the transport equation. A more detailed derivation can be found in the appendix but the result is as follows:

$$
\begin{bmatrix} \dot{w}_x \\ \dot{w}_y \\ \dot{w}_z \end{bmatrix} = \begin{bmatrix} \frac{I_{22}^v - I_{33}^v}{I_{11}^v}\omega_2\omega_3 + \left(\frac{I_{22}^c\omega_c}{I_{11}^v}\right)\omega_3 + \frac{1}{I_{11}^v}\tau_\phi \\ \frac{I_{33}^v - I_{11}^v}{I_{22}^v}\omega_1\omega_3 + \frac{1}{I_{22}^v}\tau_\theta \\ \frac{I_{11}^v - I_{22}^v}{I_{33}^v}\omega_1\omega_2 - \left(\frac{I_{22}^c\omega_c}{I_{33}^v}\right)\omega_1 + \frac{1}{I_{33}^v}\tau_\psi \end{bmatrix}
$$

Figure 5: Rotational Dynamics

As evident, the spin rate of the central cylinder will have a non-negligible impact on the dynamics of the vehicle. Normally, the spin rate would be an additional input into the system, but this requires a much more complex scheme and motor mixing algorithm than the ones presented here. The goal of this control scheme is just to stabilize the gyroscopic torques and as a first pass,

will assume a constant cylinder spin rate, allowing us to simplify the time derivative of this quantity as zero. This isn't a terrible assumption as Magnus in steady flight would ideally be spinning the cylinder at a constant high speed to take full advantage of the extra lift from the Magnus effect. However, once testing occurs and it is found that the plant does not directly align with the simulation, this term should be revisited and a proper model of the cylinder added.

It should also be noted that without a small angle approximation, the angular rate of the vehicle cannot be integrated directly to obtain the Euler angles as the Euler angles do not form an orthogonal basis. Given a rate gyro with a fast sampling rate, it is typical to employ the small angle approximation which makes this distinction trivial when implementing the controller, but in the simulation, the relationship between the time derivative of the Euler angles and angular rates is described as follows [4]:

$$\begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix} = \begin{bmatrix}1 & 0 & -s(\theta)\\ 0 & c(\phi) & c(\theta)s(\phi)\\ 0 & -s(\phi) & c(\theta)c(\phi)\end{bmatrix} \begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix}$$

Figure 6: Relationship between Angular Rates and Euler angle Rates

3.4 Linear Dynamics / Newton's Second Law

The linear dynamics can be described via Newton's second law (F=ma), where the forces being considered are gravity, a simplified linear drag model, the thrust from the motors, and the additional lift from the cylinder. The only caveat is that the thrust from the motors described in **Fig 4** is in components of the rotating body frame attached to the center of mass of the quadcopter. A key aspect of Newton's second law is that it must be taken in terms of an inertial frame, meaning that the thrust vector must be rotated into the inertial frame via the rotation matrix described by **Fig 1** before being accounted into the equation. At the time of this report, testing has not occurred to characterize the amount of lift obtained from the Magnus effect. While the theoretical lift could be approximated by the Kutta Jowaski Equation [5], the assumed relationship returns an optimistic amount of lift and values that conflict with CFD done by previous members as well. Due to this, a constant conservative lift equivalent to 1/5 the weight of Magnus in the body frame Z direction / Yaw Axis associated with a non-zero spin rate of the central cylinder was assumed. The resulting dynamics are as follows:

$$\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = \begin{bmatrix}0\\ 0\\ -g\end{bmatrix} + \begin{bmatrix}\frac{-k_d}{m}\dot{x}\\ \frac{-k_d}{m}\dot{y}\\ \frac{-k_d}{m}\dot{z}\end{bmatrix} + \left({}^{\mathfrak{B}}R^{\mathfrak{N}}\right)^T \begin{bmatrix}0\\ 0\\ \frac{F_T}{m}\end{bmatrix} + \begin{bmatrix}0\\ 0\\ \frac{F_C}{m}\end{bmatrix}$$

Figure 7: Linear Dynamics

4. Control Architectures

This section will feature a working simulation of a potential control architecture, followed by other attempts/options that may suit the future team better once it has been decided that Ardupilot is not sufficient for this use case. All the alternatives build directly off the work done here and can be swapped out once it is determined that is the route the team wants to pursue.

The main control architecture features a cascade control in an attempt to autonomously control the X Y Z of the drone without the use of a remote control and instead, with a reference X Y Z value. Six PIDs were then used to control the error of the 3 Euler angles as well as the 3 translational motions. However, this results in an extremely tedious tuning process of the PIDs, all of which are coupled with each other in some way. Additionally, despite using PIDs, the system is highly non-linear due to the thrust vector depending on the orientation of the quadcopter and doesn't allow for the use of any of the frequency domain tools (Bode Plots, Root Locus, Nyquist plots, Phase / Gain Margins).

Another promising approach and the next step in this project is to use gain scheduling, in which the variable that the controller schedules on is the pitch angle. This potentially solves the instability issue of pitching the quadcopter forward close to 90 degrees by designing multiple gains that discretize the change from 0 to 90 degrees. This also results in the plants being linear and thus allows the user to gain access to Simulink's PID autotuner tool and access to the linear quaternion schemes using LQR [4].

Lastly, a much simpler control scheme that takes in throttle, pitch, roll, and yaw commands, much like how a regular RC aircraft would, with the user being in the loop to achieve the desired translational position could easily be derived from the working architecture. The benefit of this scheme despite its simplicity is that it allows for faster iterations of gains, as the pitch, roll, and yaw PIDs are linearizable thus allowing access to the frequency domain tools for analysis by hand or Simulink's autotuner tool for on-the-fly adjustments. This setup would potentially be much more stable since the main issue with the cascade control was the instability in the Y control and the PIDs and would remove the reliance on the Ardupilot software.

4.1 Cascade Control Scheme Overview:

To achieve autonomous control of the X Y Z, the first setup utilizes a cascade control scheme. This setup lends itself very easily to the natural way a quadcopter moves in the X and Y directions as described earlier. Thus, the outer loop will have the position control, which takes in a reference X Y, and Z motion and attempts to decrease the error in those directions. The PID taking in the Z-error will return a Thrust command, while the PIDs taking in the X and Y error will return a Pitch and Roll reference. These references go into the inside loop which houses the attitude controllers and adjusts for any error in the orientation. A separate yaw reference would have to be fed into here but due to the initial goal of stabilizing the gyroscopic torques, a constant reference of 0 yaw was chosen throughout the rest of the design. The attitude controllers will return a Pitch, Roll, and Yaw command, which alongside the Thrust command from earlier

will all go into the “motor mixing algorithm”, which takes in the requested thrust and torques and returns a set of desired motor speeds according to Fig X. This all goes into a simplified actuator model and returns the motor speeds/inputs into the system/plant. An overview of the scheme is as follows:

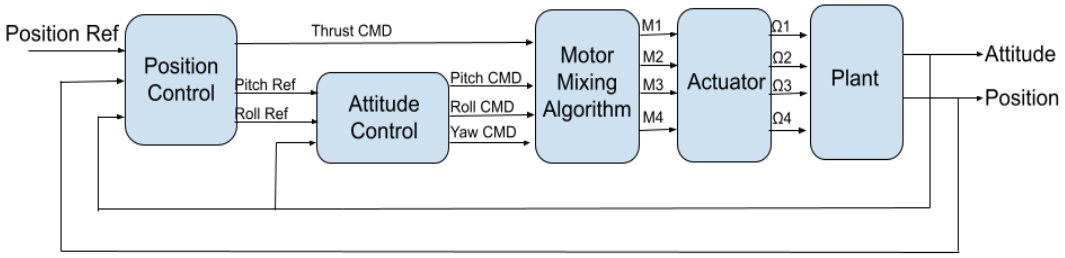


Figure 8: Cascade Control Scheme Block Diagram

4.1.1 Outer Loop / Position Control:

As mentioned, the Z direction isn’t dependent on the orientation of Magnus and is just a PID that will return a thrust command based on the Z error. The X and Y error, however, has slightly more nuance and cannot directly be used towards a Pitch and Roll reference. The X and Y errors will be in the inertial frame and must be rotated into the frame of the quadcopter via the yaw angle before being useable. This is akin to a person taking a desired amount of steps forward, versus rotating 45 degrees and taking that same amount of steps forward. Despite having done the same command, the final positions are not the same due to the orientation. Much like this example, giving the error of the X and Y in the inertial frame to the drone with a non-zero yaw axis (which will happen due to the gyroscopic torques) will result in undesired behaviors as the roll and pitch commands will be based on a zero yaw case. This is mathematically described as follows:

$$\begin{bmatrix} x_{err}^{\mathfrak{B}} \\ y_{err}^{\mathfrak{B}} \end{bmatrix} = \begin{bmatrix} c(\psi) & s(\psi) \\ -s(\psi) & c(\psi) \end{bmatrix} \begin{bmatrix} x_{err}^{\mathfrak{N}} \\ y_{err}^{\mathfrak{N}} \end{bmatrix}$$

Figure 9: Transformation of X Y Error in inertial frame to body frame

After properly obtaining the components of the X and Y error in terms of the body frame, they will go into 2 PIDs that will control the roll and pitch references. There were saturations placed on the output of the PIDs as an attempt to make the system more stable such that the controller does not request an angle that will either make it flip over, become uncontrollable, or reach a gimbal lock. This also serves as a safety factor against how unstable the system is. Since Magnus has not been flown at the time of writing this report, the amount of pitch and roll that can be achieved before the system becomes unstable has not been determined and a conservative value can be selected to start. Lastly, the roll reference was made negative due to the definition of the body frame, where a negative roll would produce a positive Y translation.

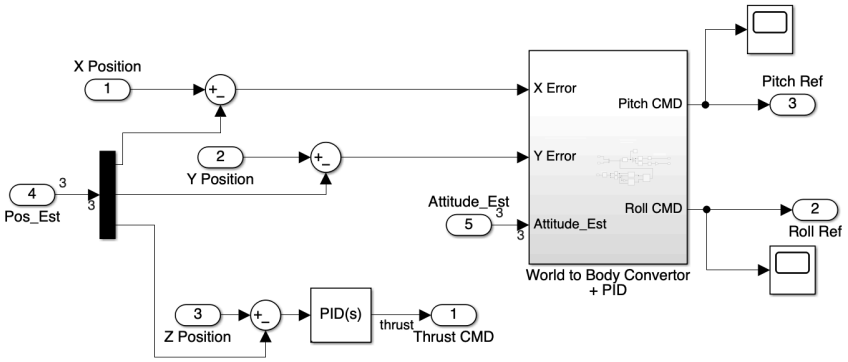


Figure 10: Position Controller (Altitude Controller)

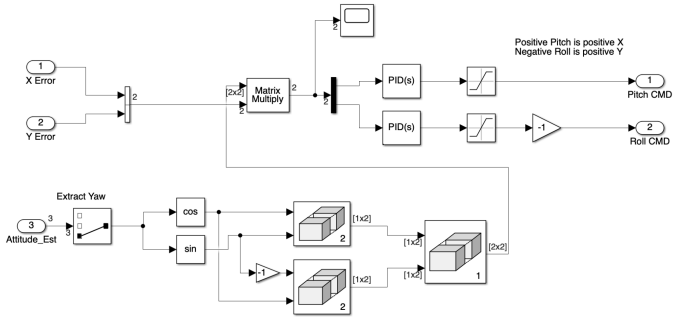


Figure 11: World to Body Converter + X Y Controller

4.1.2 Inner Loop / Attitude Control:

Since all 3 DOFs associated with the attitude are decoupled from all other states, it makes the design of the control extremely simple. This block features 3 PIDs, one for each of the Euler angles, with the roll and pitch reference being calculated from the X and Y PIDs, and the yaw reference being a user input (0 for stability). The outputs of these PIDs will be desired torque commands around the respective axis. It should be noted that as with all cascade control schemes, the goal is to get the inner loop to operate at a higher bandwidth (about 1 order of magnitude) than the outer loop to obtain a degree of separation between the two. This allows for the quadcopter to obtain the requested orientation much faster than the requested translational movement, essentially obtaining the thrust components in a quick enough manner that instability doesn't arise from a time delay.

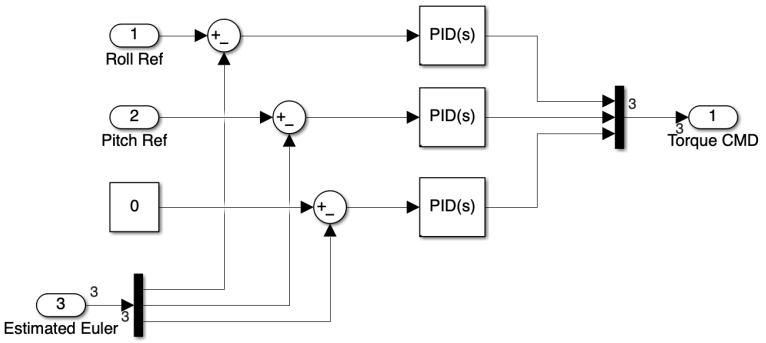


Figure 12: Attitude Controller

4.1.3 Motor Mixing Algorithm (MMA):

The “Motor Mixing Algorithm” splits up the requested Thrust and Torque commands as followed by Figure X. This algorithm assumes an equal splitting of all commands into each motor which is how conventionally quadcopters work. In the future when the center cylinder is modelled, the thrust commands will have to be split between the four motors and the center cylinder. A sigmoid function may be helpful in an initial pass at this as it allows for a seamless transition between the commands being distributed to the motors and the cylinder after passing a certain threshold.
NOTE: The gain shown is to divide the force/torque by the coefficients in front of the sum of motor speeds in Figure X but this is not needed and was left for completeness.

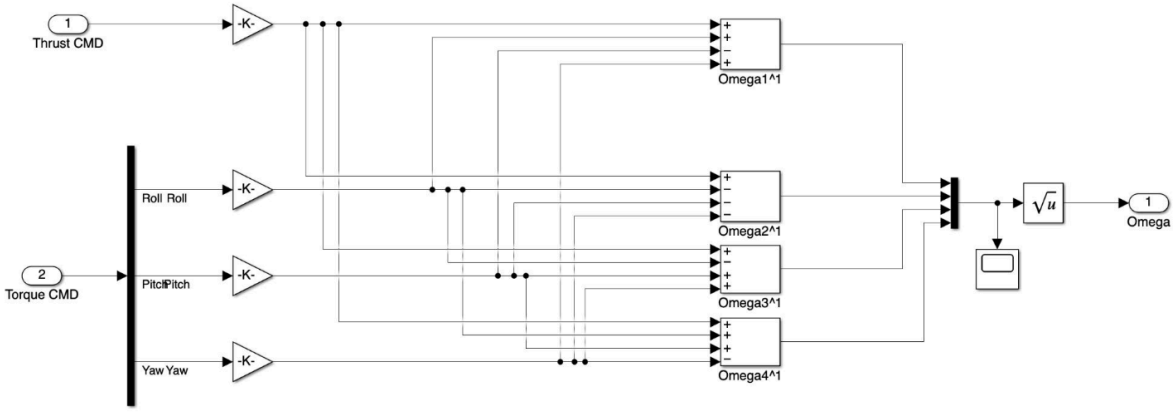


Figure 13: MMA

4.1.4 Actuator Model

As of now, the actuator just features a time delay that is modeled by a first-order transfer function. This allows for a rise time of approximately four times the selected time constant to model the inability of the motors to instantly spin up to the required spin speeds. However, the

electric motors are observed to spin up extremely quickly so the time constant used in the transfer function was not very large and is almost negligible. There is a saturation based on the max spin rates of the motor determined by the rated KV and knocked down by ~50 rad/s, resulting in a value of 650 rad/s.

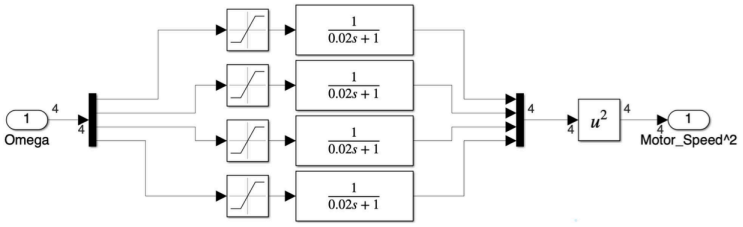


Figure 14: Actuator Model

$$\frac{Output}{Input} =$$

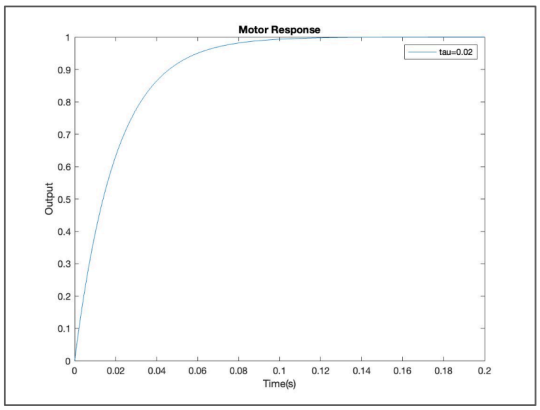


Figure 15: First-order system response with a time constant of 0.02

4.2 Gain Scheduling

As previously mentioned, gain scheduling presents a promising way to control the highly non-linear plant with linear techniques. Previous work on this with quadcopters sees the gain scheduled on the yaw angle of the quadcopter, but the results show a much more agile quadcopter than Magnus could achieve [6]. Due to this, it is proposed that the variable of choice should be the pitch angle, with zero pitch representing the stabilizing case, linearizing the attitude around zero roll and yaw angles. This however introduces the issue of controlling the roll angle once in a non-zero pitch. It is possible to schedule the gains based on a combination of both the roll and pitch angle, but this would increase the amount of cases needed to be designed. If this must be done, a number of "nodes" representing the pitch and roll angles can be selected and gains designed for those points, where the gains are then interpolated between the nodes if a pitch and roll angle falls in between the values. The max roll angle to be designed around should be tiny (~$\frac{\pi}{8}$-$\frac{\pi}{16}$ radians) such that any roll maneuvers are tiny and wouldn't cause the system to fail in such an unstable state.

Gain scheduling for this application is extremely time intensive due to the number of PIDs that must be tuned and tested before implementing the next, which was why this wasn't simulated. With the possibility of the work shifting more towards simulation, this would be a good path to pursue if more than one person is working on the control implementation or just a

good exercise to get familiar with the system. At this point, it would also be helpful to reintroduce quaternions to represent the attitude, as it would bypass any gimbal-lock issues and be more computationally efficient when onboard. The dynamics would remain the same, with the only change being the addition of a quaternion integration to propagate the attitude from the rotation rates. This also changes the rotation matrix used to rotate the thrust from the body frame to the inertial frame, which should remain unchanged if corrected properly. A definition of quaternion kinematics from the Markley textbook is presented in the appendix [X]. Once a sufficient amount of gains have been designed for the 6 PIDs, a switching mechanism can be implemented into the non-linear simulation directly while keeping the cascade control scheme.

4.2.1 Linearization of Dynamics

The attitude plant (inner loop) will be linearized about 0, such that $\omega_{1,eq} = \omega_{2,eq} = \omega_{3,eq} = 0$. This presents the case in which we are in a stable configuration in attitude, making it the most natural point to linearize about.

$$\begin{bmatrix} \delta\dot{\omega}_1 \\ \delta\dot{\omega}_2 \\ \delta\dot{\omega}_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & \frac{I_{22}^c}{I_{11}^v}\omega_c \\ 0 & 0 & 0 \\ -\frac{I_{22}^c}{I_{33}^v}\omega_c & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta\omega_1 \\ \delta\omega_2 \\ \delta\omega_3 \end{bmatrix} + \begin{bmatrix} \frac{1}{I_{11}^v} & 0 & 0 \\ 0 & \frac{1}{I_{22}^v} & 0 \\ 0 & 0 & \frac{1}{I_{33}^v} \end{bmatrix} \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix}$$

Figure 16: Linearized Attitude Dynamics

While the linearization of the translational dynamics can be for both the pitch and roll axis, for now, only the pitch angle will be considered. Since the pitch equilibrium angle we want to consider isn't necessarily zero, it can take on any value and thus will be left as a variable, with the roll and yaw angles set to zero. The rotation matrix from Fig. X then simply becomes a rotation about the pitch axis.

$$\left[{}^{B}R^{I}\right] = \begin{bmatrix} cos(\theta_{eq}) & 0 & -sin(\theta_{eq}) \\ 0 & 1 & 0 \\ sin(\theta_{eq}) & 0 & cos(\theta_{eq}) \end{bmatrix}$$

Figure 17: Rotation Matrix for Pure Pitch

Since the rotation matrix was the only nonlinearity from the original translational dynamics, the equations **Fig 7** can be compactly defined as:

$$\dot{\mathbf{x}} = \begin{bmatrix} 0_{3x3} & I_{3x3} \\ 0_{3x3} & -\frac{k_d}{m} I_{3x3} \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{-sin(\theta_{eq})}{m} & 0 & 0 \\ 0 & 0 & 0 \\ \frac{cos(\theta_{eq})}{m} & \frac{1}{m} & -1 \end{bmatrix} \begin{bmatrix} F_T \\ F_c \\ g \end{bmatrix}$$

Figure 18: Linearized Translation Dynamics

Where **x** represents the state vector containing the position and velocity.

Using these linearized plants, several controllers can be designed for the cases mentioned above. Only a couple likely have to be designed for the attitude dynamics, since the cascade control scheme should still be used for this, meaning all the assumptions about the inner loop hold. This limits the inner loop to have an order of magnitude in separation from the bandwidth of the outer loop such that they are decoupled properly, making it one of the main considerations when designing specific gains for chosen pitch/roll angles.

4.3 Human in the loop operation

As a final application of the cascade control scheme from section 4.1, the X and Y controls can be removed, with the only feedback loop being the altitude as shown below:

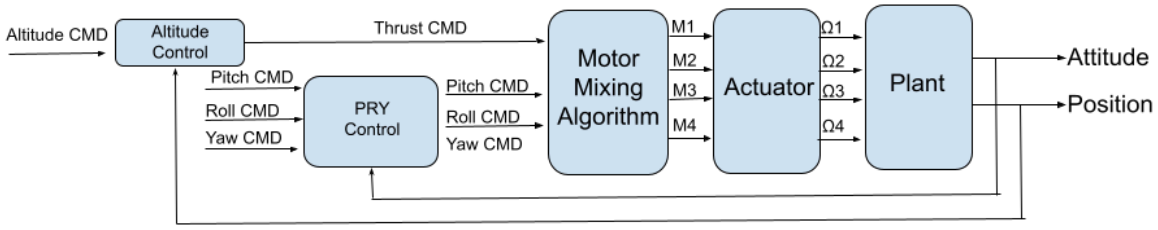


Figure 19: Open Loop Set-up

This is useful as it allows for the quadcopter to maintain a stable altitude while allowing the user to directly input pitch, roll, and yaw CMD with the remote controller. As mentioned before, controlling pitch and roll allows for indirect access to controlling the X and Y directions. By having the user in the loop feeding these commands, it removes the reliance on a feedback loop without a great sense of robustness, potentially allowing for better control than an automatic method could achieve. This also offers a better alternative to Ardupilot as it strips down all the excess features, allowing for better customization and access to the different parameters. For example, while there is no feedback control in the PRY controls, there could be associated saturations or fail-safes based on observations during flight tests. Achieving a stable altitude and being able to control the PRY via remote control was the main goal for the prototype this semester and the reason why Ardupilot became a large focus.

5. Simulation Results

Due to the amount of dependence on the Cascade controller scheme for future changes/variations, that was the focus of a proof of concept for simulation in Simulink.

5.1 Simulation Parameters

Within the simulation, there were 8 main parameters that had been selected prior to tuning the gains. These 8 parameters included the following:

1) The moment of inertia matrix of the entire vehicle (including the central cylinder)
2) The moment of inertia matrix of just the central cylinder
3) The total mass of the vehicle
4) The two different moment arms to the motor
5) The thrust constant of each motor
6) The torque constant of each motor
7) A motor time constant
8) Drag coefficient

Since an accurate CAD with accurate mass properties for the system was not made, the first three parameters had to be estimated. The moment of inertia for the central cylinder was estimated by assuming two dense disks oriented such that the pitch axis went through each of their respective centers. The moment of inertia of one disk oriented as such was calculated in its principle frame such that no off-diagonal components exist, with the parallel axis theorem then applied to using a measured distance from the center of the prototype. This was then multiplied by two to represent the symmetry (can do a parallel axis with the other disk and add them to obtain the same results).

The moment of inertia of the entire vehicle was estimated by assuming a dense spherical object at the center, followed by four motors at each of their respective corners which were also modeled as disks. This required the assumption that the majority of the center mass was from the battery with an extra factor multiplied to account for the carbon rods and other 3-D printed components. The same approach from before was followed. First, the principle moment of inertia was calculated for the sphere and disks representing the motors. Next, the parallel axis theorem was applied to each of the four motors to obtain their inertia relative to the center point. Finally, to obtain the final estimate of the moment of inertia of the vehicle, the moment of inertia of each of the four motors following the parallel axis theorem, the dense sphere, and the calculated central cylinder moment of inertia from before were summed.

The total mass of the vehicle was estimated prior to any large changes to it and was just the sum of all the assumptions from the moment of inertia calculations. The moment arms from the motor to the pitch / roll axis were measured directly from the vehicle, with there being a distinction since the motors are not equidistant from both axes and thus will produce a different amount of thrust. The thrust constant of a motor was estimated via a thrust curve provided by the manufacturer as well as the KV from the website. The provided thrust curve is below with the X axis representing the current and the Y axis representing the thrust in gram-force:

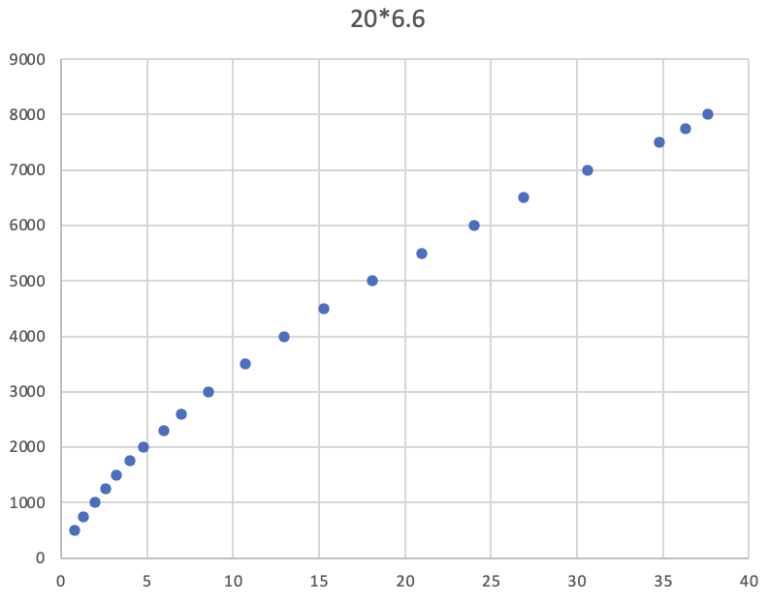


Figure 20: Thrust vs Current Curve

This shows a max Thrust of about 8000 gf, which is 78.4532 Newtons. Using the KV rating, which provides a linear relationship between voltage and rotation speed, a max rotation speed from the motors in RPM can be found by taking the max voltage of the battery and multiplying it by this number. The battery provides a voltage of 22.2V, and the motors provide a KV of 300, resulting in a max spin of 6660 RPM, which translates to around 697 rad/s. However, to account for any degradation and to avoid hitting this upper limit in an effort to prevent overloading the ESCs, this was lowered to 650 rad/s for saturation purposes as well as for future calculations. Recall from Fig X, that the relationship between thrust and the square of the motor speed is assumed to be linear, so a rough approximation of the thrust constant can be obtained from the slope. Using the 0 thrust 0 rotation speed point, as well as the knowledge that the max thrust is about 78.4542 and the max speed squared is 422500, a slope of about 0.000185 was found to be the thrust constant. While the torque constant is related to the fluid mechanics of the interaction between the blades and the oncoming airflow, this slope was multiplied by the moment arm to obtain an estimated value.

As mentioned prior, the motor time constant is pretty hard to find and was just assumed to be about 0.02 to achieve a 0.1-second response time. Lastly, the drag coefficient was assumed to be a low number as the prototype will not be going at great enough speeds for drag to play a large factor. A potential solution to obtain this coefficient is by running an Extended Kalman filter that is able to estimate this parameter directly.

Table 1: System Parameters

I^v	I^c	$m_{vehicle}$ (kg)	L_ϕ,L_θ (m)	k	b	τ_m (s)	k_d
diag([1.408; 0.505; 1.67])	diag([0.37; 0.24; 0.37])	10	0.35,0.7	1.85e-4	1.3e4	0.02	0.005

5.2 Tuning Process

Tuning for the PIDs should begin at the inner loop. Due to the cascade control's objective of decoupling the inner and outer loop, a separate feedback loop such that only the attitude is looked at can be formed for the purpose of tuning as shown below:

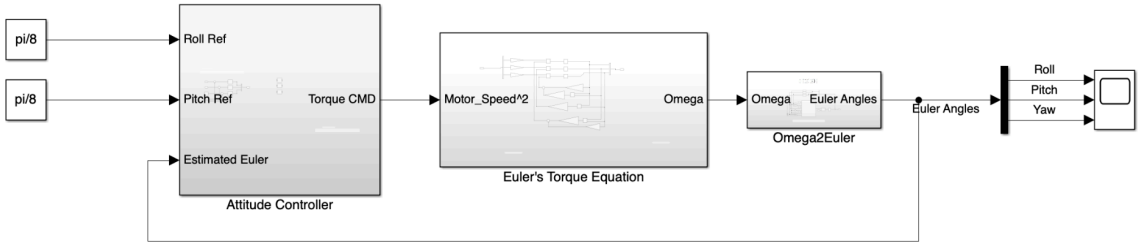


Figure 21: Decoupled Attitude Loop

The benefit of separating this comes from Simulink's autotuner tool, which allows you to tune the PIDs based on the desired system response. In the case of the larger system, Simulink sometimes runs into issues linearizing the Euler equations due to different saturation blocks and factors. For the sake of this proof of concept, an extremely fast but stable system behavior from the attitude loops was desired, which was achieved by either maxing out the bandwidth and phase margin options under the Frequency domain or choosing the fastest response time and most robust transient behavior if the Time domain was selected. An example of the PID controlling pitch is seen below:

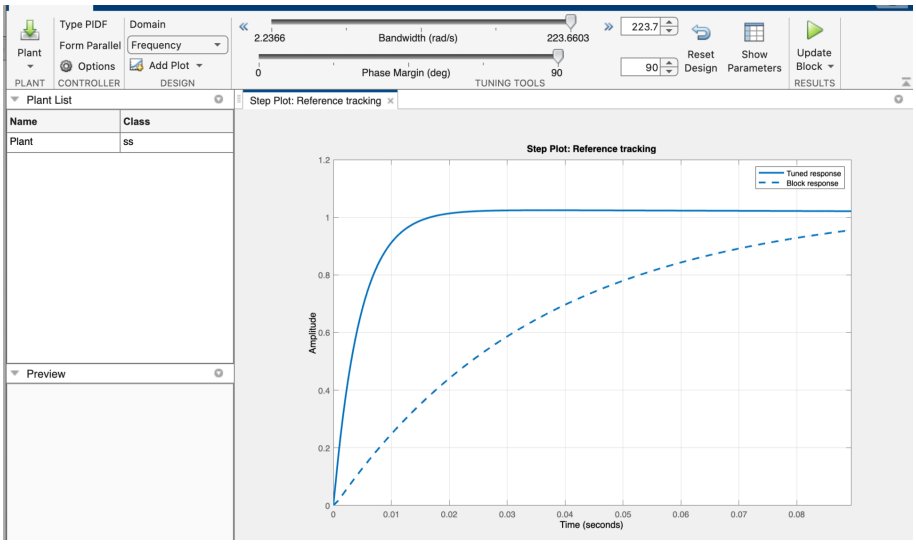


Figure 22: Simulink PID Tuner Tool

While this is highly unrealistic, it serves the purpose of decoupling the inner loop while maintaining stability. Future work can be done to make this response more realistic and match the actual system. With the inner loop tuned via this method, the 3 PIDs can be directly transferred into back into the non-linear loop.

Next, for the outer 3 PIDs, the same method could not be used due to the force being rotated by the attitude before being added onto the acceleration terms, resulting in non-linearities that Simulink could not get around. Instead, the convention of the Ziegler - Nichols tuning method was followed. This method has the proportional gain tuned and increased until the system showed instability in the step response. Once this point is reached, the proportional gain is lowered and the integral and derivative gain are set to some proportion of this amount. While this method could not be applied directly to this system, it was found that tuning the proportional gain to instability and then adding derivative gain to stabilize before adding an integral term decreases the steady-state error worked best.

With the altitude / Z command being the most decoupled from the rest, that was the PID that was tuned first. This process was relatively quick with only a step response into the Z. Next, the X PID was tuned with a step response into both the Z and X. At this point, the method above could be used as a starting point but the gains largely had to be tuned intuitively to obtain the desired characteristics in both the Z and X while stabilizing a reference in the X. Lastly, the most tedious part comes from tuning the Y. Due to the cylinder, the quadcopter is extremely unstable when trying to roll, which unfortunately is the maneuver needed to control the Y. There was no good method of trying to tune this except for using the X PID gains as a starting point. The goal when tuning was to see the magnitudes of all 3 X Y Z motions and attempt to shrink them. If a change to one of the gains of Y results in a larger deviation from a step of 1, then it is likely that is not the way to go and to try to decrease it. This would go on until the error is within a controllable amount (~10m). When this is reached, the gains in the X and Z can be returned in an attempt to further drive the error down. Only small changes to the gains should be made at this point. When the 3-step responses are stable, the PIDs should be checked such that a step in the Z, a step in the Z + X, and a step in the Z + Y, are all still stable.

5.3 Simulation

After the gains that stabilized the states were selected, a couple of requirements for a step response were chosen to further tune the gain. These included:

- Rise time <20 seconds
- Overshoot <20%
- Settling time <40 seconds

These were selected such that the quadcopter is somewhat agile and wouldn't waste extra control effort in trying to correct any overshoot. These goals were also achievable at a conservative level in order to focus on this being a proof of concept rather than a well-tuned system ready to be implemented and could become more strict once there is more confidence in the system parameters.

5.3.1 Step response in X Y Z

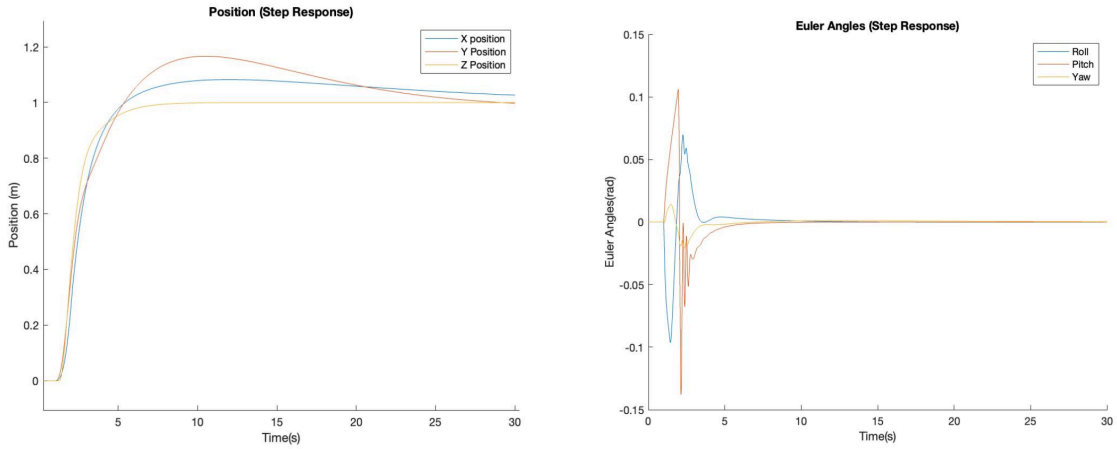


Figure 23: Tuned System Step responses

As can be seen, the system was able to respond well to a step response in the X-Y-Z. There is a slight overshoot in the X-Y but this is less than the 20% margin set. The settling time was met on the X Y and Z, although the X has a slightly longer settling time. Lastly, all three met the rise time requirements by a significant margin. Looking at the Euler angles plot, the attitude also reacts as expected. As the quadcopter is translating in the X and Y, the roll and pitch changes in order to point the thrust vector to achieve this. Then, as the error in the X and Y start to decrease, the commanded roll and pitch also start to decrease to 0, resulting in the settling of all three Euler angles about 0. As mentioned before, the yaw angle is always set to 0 to demonstrate counteracting the gyroscopic torques so this was also successful. The only slight concern is in the fast oscillation of the pitch angle, which corresponds to the X command. While no definitive answer is known, this is most likely due to the pitch PID resulting in a slight overshoot, leading to the oscillatory response once put into the full system. However, the oscillation magnitudes are less than 0.1 radians, which corresponds to roughly 5.73 degrees, which is a small enough amount that wouldn't lead to any major instabilities.

5.3.2 Ramp response in the Z + Step X Y

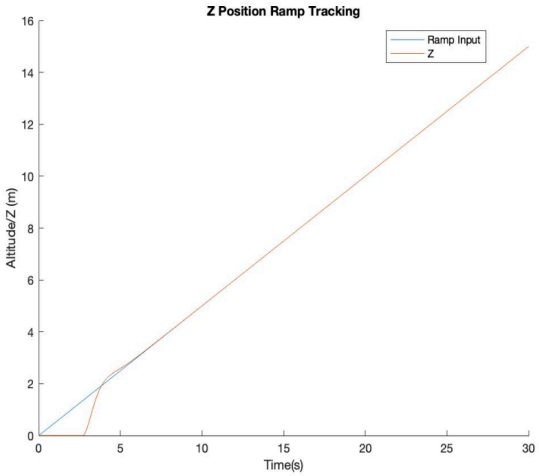


Figure 24: Ramp Input in Z

Typically with a quadcopter, rather than give a step response, the throttle either ramps up or down. Thus, the control scheme was also tested to see how well it would react to a ramp input in the Z in case a more gradual climb was desired. Based on the first graph, it can be seen that after an initial delay in the error building up, the system is able to track the response extremely well. This was further tested with an additional step input in the X and Y.

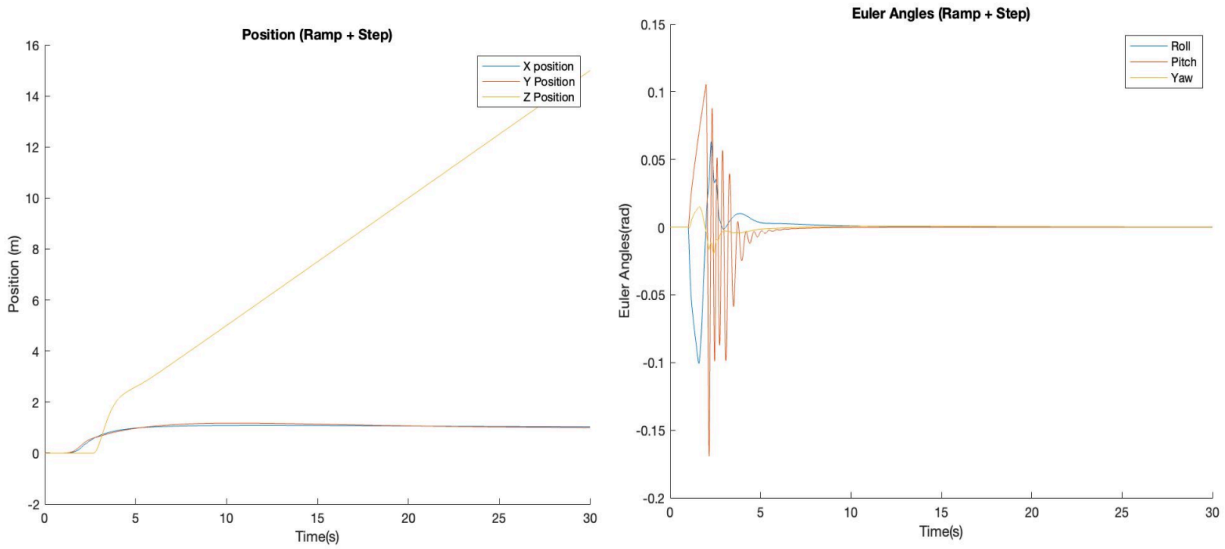


Figure 25: System Response to Ramp + Step Inputs

This setup also saw success with stable responses. It should be noted that the quadcopter begins to translate in the X and Y before lifting off the ground, which isn't realistic but is due to not properly modeling the ground / adding cases to prevent this. Instead, this should be seen as the quadcopter in a slight hover, thus ignoring any interaction with the ground. This helps make this response more realistic while not taking away from the results. Additionally, there are more oscillations in the pitch axis but once again, this doesn't lead to any overall instability so it's assumed that these could be tuned out if necessary.

6. Impacts from Flight Test

During the second semester, the focus switched from simulation work to integration and getting the prototype working with Ardupilot / via the remote control. This was important as seeing how the system reacts to certain inputs from the user or maneuver, in general, gave insight into how it should react in simulation as well as what should be recreated within the simulation. For example, when throttling the thruster, there was a longer delay in the overall system response even when the throttle was at a specific point. This indicates that while the 0.02 time constant of the actuator might be a good assumption, there is a further system delay from the produced thrust to the actual response that isn't modeled. Additionally, it was noticed that the quadcopter was unstable in the pitch angle. At first, this was theorized to be the case due to the moment arm associated with this axis being shorter but it was validated when a slight non-zero pitch command led to the drone flipping forward. More insight was hoped to be gained about the agility of the quadcopter from these flight tests, but unfortunately, each one suffered setbacks that resulted in the test being cut short.

During the flight tests, the Stablize mode was also tested out. It was noted that this mode significantly improved the throttle control, stabilizing the quadcopter around zero pitch roll. This could easily be implemented via section 4.3 and then commanding a constant 0 pitch and roll, validating the usefulness of such a control scheme. Additionally, due to the instability regarding auto mode, if future work is to be done into implementing an in-house controller, the one in section 4.3 should be set up first, as many of Ardupilot's useful modes could be derived from such a scheme, including the Stablize mode as described above, alt-hold, and even loiter, which is a combination of alt-hold and stabilize.

Lastly, while multiple attempts have been made to get Ardupilot's auto mode to work, there have been consistent issues. It is currently theorized that auto mode needs much further tuning, which would require the quadcopter to be able to fly autonomously, to begin for this to work. While more attempts to get this working should be made, the implementation of 4.3

7. Conclusion / Future Work

In conclusion, while the implementation of a control scheme onboard could not be achieved due to time constraints and prioritizing getting the physical prototype to work, a significant amount of progress has been made on the simulation and modeling side. This included researching and modeling the dynamical equations, the literature review of different control schemes, and investigations of control techniques that spanned both linear and non-linear systems. This lays the foundation for many of the future works with the control system that could be done with the project if this route is pursued any further. A Cascade Control scheme was also shown to be successful in stabilizing the dynamics of the quadcopter with estimated system parameters. During the second semester, my work regarding the integration was not discussed in this report in order to keep it focused on the control system, and the overall impact is discussed in my teammate's report.

From here, there are two main steps forward. On the simulation and modeling side, it is recommended that a gain scheduling approach be investigated, as mentioned in section 4.2. This method sees the potential of translating the autonomy of the cascade control scheme into the tiltrotor configuration that is desired. It also simplifies the tuning process immensely by presenting fully linear models and thus allows for a quantification of the robustness of each gain chosen. The switching between each controller could be discrete, such that once the variable passes set values, the controller would switch, or if the gains are "smooth" between each design case, the gains could also be interpolated in between. This method would present a smoother system response as the gains are now parameterized by the scheduled variable but is harder to ensure stability of the gains in between.

On the implementation side, the full 6DOF control should be forego for the one presented in 4.3. This is because tuning the X and Y gains was extremely tedious and time-consuming with no real indication of robustness, an issue that will only be exaggerated when considering real-world disturbances and sensor noises. Instead, 4.3 allows for semi-autonomy by either having the user command a thrust / PRY via the remote controller, or having a set input into them. This is set up is extremely useful as it allows for the recreation of almost all of Ardupilot's

flight modes with much more customization and the ability to actually model the impacts of changing parameters on the system response. As of now, this ability is not available, causing a blind reliance on Ardupilot's built-in autotuner as well as any other generic method of changing parameters.

As for recommendations, it would be extremely helpful to have the future team decide early on whether the implementation of a control system aside from Ardupilot is to be implemented and to recruit more people to make it work. Having one person work on the initial control system required a balance between contributing towards prototype discussions while spending extensive time working on an independent project. There was a disconnect between the two during this year due to the prototype having never flown before and many of the system parameters being uncertain, one of the main ones being the characterization of the thrust of the central cylinder. This resulted in the implementation taking a backseat when the integration work started to ramp up to attempt to speed up the process as it was decided the risk and time put into getting a controller working didn't align with the team's goals. If a member of the team is interested in this work, I'd recommend taking advantage of the minidrone in the lab to conduct testing of the controller. This assumes that enough manpower is allocated to collecting data to characterize the thrust such that this extracurricular project wouldn't interfere with the rest of the team. While the gains or parameters used would not be able to be transferred over to the full scale, it would allow for the team to gain knowledge on the implementation portion, including the code structure, electronics required, and different fail-safes not modeled on Simulink that is encounter during testing.

8. Bibliography

[1] Kang, Joo-Won, Quaternion-based Dual loop Nonlinear Trajectory Control of Quadrotors
URL:
https://repository.gatech.edu/entities/publication/752dc2c9-90ff-4c1b-bb89-c7b88e3f4261

[2]Emil Fresk; George Nikolakopoulos, “Full quaternion based attitude control for a quadrotor”
URL:
https://ieeexplore.ieee.org/document/6669617

[3] Benjamin Morrell, Marc Rigter, Gene Merewether, Robert Reid,
Rohan Thakker, Theodore Tzanetos, Vinay Rajur and Gregory Chamitoff1 “Differential Flatness Transformations for Aggressive Quadrotor Flight”
URL:
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8460838

[4] Andrew Gibiansky, “Quadcopter Dynamics & Simulation”
URL:
https://andrew.gibiansky.com/blog/physics/quadcopter-dynamics/#:~:text=Equations%20of%20Motion&text=where%20%CF%89%20is%20the%20angular%20velocity%20vector%2C%20I%20is%20the,at%20the%20end%20of%20each

[5] NASA, “Approximate Lift on Spinning Ball”
URL:
https://www.grc.nasa.gov/www/k-12/VirtualAero/BottleRocket/airplane/beach.html#:~:text=The%20Kutta%2DJoukowski%20lift%20theorem%20states%20the%20lift%20per%20unit,determine%20the%20strength%20of%20rotation.)

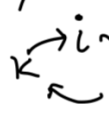
[6]Sawyer, Shaun, “Gain Scheduled Control of a Quadcopter UAV”
URL:
https://uwspace.uwaterloo.ca/handle/10012/9488

9. Appendix

9.1. Quaternion Kinematics

Quat. Kinematics (Markley A.6)

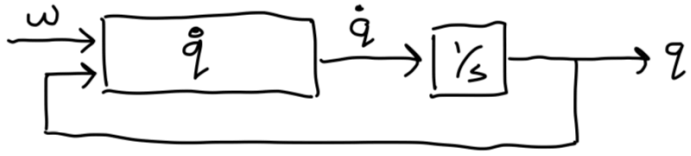
$$\dot{q} = \frac{1}{2}\omega \otimes q \quad \text{(state)}$$



where $i \times i = j \times j = k \times k = -1$

$$= \frac{1}{2}\begin{bmatrix} \omega_1 i \\ \omega_2 j \\ \omega_3 k \\ 0 \end{bmatrix} \otimes \begin{bmatrix} q_1 i \\ q_2 j \\ q_3 k \\ q_4 \end{bmatrix} = \omega_1 i \times (q_1 i + q_2 j + q_3 k + q_4) + \omega_2 j \times (q_1 i + q_2 j + q_3 k + q_4) + \omega_3 k \times (q_1 i + q_2 j + q_3 k + q_4)$$

$$= \frac{1}{2}\begin{bmatrix} (\omega_1 q_4 + \omega_2 q_3 - \omega_3 q_2) i \\ (-\omega_1 q_3 + \omega_2 q_4 + \omega_3 q_1) j \\ (\omega_1 q_2 - \omega_2 q_1 + \omega_3 q_4) k \\ (-\omega_1 q_1 - \omega_2 q_2 - \omega_3 q_3) \end{bmatrix}$$



9.2. Attitude Dynamics Derivation

$$h = I\omega$$

$$h_B = h_V + h_C = I_V \omega_{B/V} + I_C \omega_{V/C}$$

$$\dot{h}_B = \tau_{ext} = \dot{h}_V + \dot{h}_C = I_V \dot{\omega}_{B/V} + \omega_{B/V} \times (I_V \omega_{B/V}) + I_C \dot{\omega}_{V/C} + \omega_{B/V} \times (I_C \omega_{V/C})$$

(assume small)

$$\dot{\omega}_{B/V} = I_V^{-1}\left(\tau_{ext} - \omega_{B/V} \times (I_V \omega^{B/V} + I_C \omega_{V/C})\right)$$