Systems Theory Lab Tailsitter SP21 Independent Study

Daniel Cherenson

May 17 2021

Contents

1 Introduction

This project aims to develop and validate a working MATLAB/Simulink simulation of the tailsitter to use as a test for the learning method optimal control synthesis. The simulation needs to have equations of motion that are symbolically accurate but not necessarily numerically accurate, i.e., certain parameter values (lift coefficient, thrust coefficient, etc) can be crudely estimated as long as the equations using them are correct.

At first, the simulation only modeled the dynamics around the hover equilibrium, but near the end of the semester, a working model of forward flight dynamics was implemented.

2 Modelling

2.1 Reference Frames

The body frame axes are defined as:

- The positive x axis points out the nose
- The positive y axis points out the right wing
- The positive z axis points out the bottom of the vehicle

When the tailsitter is hovering, the positive x axis points towards the sky (up).
The Earth frame axes (NED) are defined as:

- The positive x axis points North
- The positive y axis points East
- The positive z axis points Down

Wind frame axes are defined as:

- The positive x axis points in the direction of the freestream velocity
- The positive y axis points out the right wing
- The positive z axis is the resulting right-handed orthogonal axis

The wind frame in this simulation will be the body frame rotated by the angle of attack, α, about the y axis.

2.2 State-Space Model

The inputs to the tailsitter are flap deflections f_l and f_r and two propeller speeds n_l and n_r.

States we will use:

- Position vector of the center of mass with respect to the Earth:
  $\mathbf{X}_{NED} = \begin{bmatrix} x & y & z \end{bmatrix}^T$
- Velocity vector of the center of mass with respect to the vehicle:
  $\mathbf{V} = \begin{bmatrix} u & v & w \end{bmatrix}^T$
- Quaternion describing orientation (or attitude) with respect to the Earth:
  $q = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T = \begin{bmatrix} q_0 & \mathbf{q} \end{bmatrix}^T = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$
- Angular velocity vector: $\boldsymbol{\omega} = \begin{bmatrix} P & Q & R \end{bmatrix}^T$

We will use quaternions to describe the orientation of the tailsitter. The alternative method is to use Euler angles which are more intuitive, but result in a singularity at a pitch angle of $\pm\frac{\pi}{2}$ which is the orientation of the tailsitter in hover. Using quaternions avoids this singularity and is easy to implement mathematically and computationally in MATLAB.

2.3 Equations of Motion

2.3.1 Velocity Differential Equation

This will give us a differential equation in the form $\dot{\mathbf{V}} = f(\mathbf{V}, q, \boldsymbol{\omega}, \mathbf{F})$ where $\mathbf{F}$ is the force that acts on the vehicle center of mass.

Applying Newton's Second Law, we equate the sum of forces to the mass times acceleration, or the derivative of velocity in the Earth frame (subscript E). Rigid body dynamics shows how to convert this to the body frame (subscript V), given the following equation which assumes constant mass:

$$m\dot{\mathbf{V}}_E = m\dot{\mathbf{V}}_B + \boldsymbol{\omega} \times m\mathbf{V}_B = \sum \mathbf{F} \tag{1}$$

We do this because the accelerometer measures accelerations in the body frame.

Rearranging and splitting up the forces, you get:

$$\dot{\mathbf{V}} = -\boldsymbol{\omega} \times \mathbf{V} + \frac{1}{m}(\mathbf{F}_G + \mathbf{F}_A + \mathbf{F}_P) \tag{2}$$

which has gravitational, aerodynamic, and propulsive forces on the right side.

Gravity is easily defined in the Earth frame as $\begin{bmatrix} 0 & 0 & mg \end{bmatrix}^T$. mg is positive because z points downwards in the Earth frame. To convert from the Earth to body frame for the velocity equation, we must multiply by a rotation matrix called the direction cosine matrix, which is a function of the quaternion describing orientation. This conversion is one-to-one, meaning there is a unique rotation matrix that describes rotation by a quaternion. The equation for this matrix is below [2]:

$$\mathbf{T}_{E\to B}(q) = \begin{bmatrix} (q_0^2 + q_1^2 - q_2^2 - q_3^2) & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 - q_0q_3) & (q_0^2 - q_1^2 + q_2^2 - q_3^2) & 2(q_2q_3 + q_0q_1) \\ 2(q_1q_3 + q_0q_2) & 2(q_2q_3 - q_0q_1) & (q_0^2 - q_1^2 - q_2^2 + q_3^2) \end{bmatrix} \tag{3}$$

Plugging in $\mathbf{F}_G = \mathbf{T}_{E\to B} \begin{bmatrix} 0 & 0 & mg \end{bmatrix}^T$, we get:

$$\dot{\mathbf{V}} = -\boldsymbol{\omega} \times \mathbf{V} + \mathbf{T}_{E\to B} \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + \frac{1}{m}(\mathbf{F}_A + \mathbf{F}_P) \tag{4}$$

2.3.2 Position Differential Equation

We want to track position in the Earth frame so we can command a setpoint such as a certain altitude or to follow a path. To do this we need to integrate the velocity, but first it must be transformed into the Earth frame. We will use the inverse of the direction cosine matrix $(\mathbf{T}_{E\to B})^{-1}$. This matrix is orthonormal, so its inverse is its transpose, meaning $\mathbf{T}_{B\to E} = (\mathbf{T}_{E\to B})^{-1} = (\mathbf{T}_{E\to B})^T$. The position differential equation is simply:

$$\dot{\mathbf{X}}_{NED} = (\mathbf{T}_{E\to B})^T \mathbf{V} \tag{5}$$

2.3.3 Angular Velocity Differential Equations

The derivation for angular velocity is similar to velocity. We are using Newton's Second Law for moments and angular acceleration with the same correction for the body frame:

$$\mathbf{J}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} = \sum M \tag{6}$$

$\mathbf{J}$ is the inertia matrix which we assume to be constant: $\begin{bmatrix} J_{xx} & -J_{xy} & -J_{xz} \\ -J_{yx} & J_{yy} & -J_{yz} \\ -J_{zx} & -J_{zy} & J_{zz} \end{bmatrix}$ It is symmetric because $J_{xy} = J_{yx}$ and the same applies for the other off-diagonal terms. The off-diagonal entries of this matrix relate to the symmetry of the tailsitter. Since the tailsitter is symmetric about the xz plane (left-right symmetry), the J_xy and J_xz terms

go to zero, meaning we get a reduced matrix: $\begin{bmatrix} J_{xx} & 0 & -J_{xz} \\ 0 & J_{yy} & 0 \\ -J_{zx} & 0 & J_{zz} \end{bmatrix}$.
Something to consider is a further assumption that the tailsitter is symmetric in the xy plane as well, which means top and bottom symmetry. This would make the inertia matrix a diagonal matrix and simplifies the equations.

The moments on the tailsitter are aerodynamic and propulsive. There is no gravitational moment because gravity acts at the center of mass. Rearranging the equation, we get:

$$\dot{\boldsymbol{\omega}} = \mathbf{J}^{-1}[-\boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} + \mathbf{M}_A + \mathbf{M}_P] \tag{7}$$

2.3.4 Quaternion Differential Equation

Finally, we have the differential equation for the orientation state. From [2], the quaternion differential equation is a function of q and $\boldsymbol{\omega}$, and is given as:

$$\dot{q} = \frac{1}{2}\begin{bmatrix} 0 & P & Q & R \\ -P & 0 & R & -Q \\ -Q & -R & 0 & P \\ -R & Q & -P & 0 \end{bmatrix} q \tag{8}$$

We have to initialize the quaternion, and we can get this from the Euler angles of a hover. MATLAB has a built in function called *eul2quat* which does this transformation, but I will define it here so we understand what it is doing [2]:

$$q = \begin{bmatrix} \cos\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2} \\ \sin\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} - \cos\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2} \\ \cos\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} \\ \cos\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} - \sin\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} \end{bmatrix} \tag{9}$$

For example, if we start in a hover, $\phi = 0$, $\theta = \frac{\pi}{2}$, and $\psi = 0$. Plugging this into the above equation, we get $q = \frac{\sqrt{2}}{2} + 0\mathbf{i} + \frac{\sqrt{2}}{2}\mathbf{j} + 0\mathbf{k} = \begin{bmatrix} \frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} & 0 \end{bmatrix}^T$

2.4 Propulsive Forces and Moments

The propeller thrust force equation for one motor is given below:

$$T = C_T \rho D^4 n^2 \tag{10}$$

where C_T is a non-dimensional coefficient of thrust, ρ is the air density, D is the propeller diameter, and n is the motor speed in Hz. The value of C_T would be determined empirically, and might not be constant, but for this simplified model it will suffice. Since each motor speed is actuated separately, the left and right thrusts are added together to form the propulsive forces:

$$F_P = \begin{bmatrix} T_l + T_r \\ 0 \\ 0 \end{bmatrix} \tag{11}$$

This is the only propulsive force since the propellers lie along the body x-axis.

Using the distance from the propeller to the aircraft centerline d as the moment arm, the propulsive yaw moment is $N_P = d(T_l - T_r)$. The left motor contributes to a positive yaw moment and the right motor contributes to a negative yaw moment.

The propellers are counter-rotating, so when they have the same speed, the reaction torques on the airframe cancel out. When the motors have different speeds, this results in a propulsive roll moment. For this moment, start with the equation for power and then divide by angular velocity of the blades to get torque: $\tau = \frac{P}{2\pi n} = \frac{C_P \rho D^5 n^3}{2\pi n} = \frac{C_P \rho D^5 n^2}{2\pi}$. C_P is the coefficient of power which is also determined empirically. The difference between the right and left motors is then propulsive roll moment: $L_P = \tau_r - \tau_l$. The top of each motor moves towards the center of the aircraft, so that means the left motor turns counterclockwise when viewed from the front. The torque on the aircraft from the propeller is clockwise, which is a negative roll moment, so the total moment is the right torque minus the left torque.

Thus the total propulsive moment is:

$$\begin{bmatrix} L_P \\ M_P \\ N_P \end{bmatrix} = \begin{bmatrix} \tau_r - \tau_l \\ 0 \\ d(T_l - T_r) \end{bmatrix} \tag{12}$$

where τ and T are functions of both motor speeds.

2.5 Aerodynamic Forces and Moments

2.5.1 Important Variables and Considerations

An important variable in the aerodynamics is the freestream velocity V_∞, which is the total velocity of the air hitting the wings which will produce lift. It is simply the magnitude of the body velocities $V_\infty = \sqrt{u^2 + v^2 + w^2}$. The freestream velocity also defines the angle of attack, α, which is the angle between the oncoming air, V_∞, and the body x-axis, or the nose. It is defined as $\alpha = \arctan(\frac{w}{u})$. All aerodynamic forces (and the moments that the forces create) are dependent on angle of attack.

Another important variable is the velocity of the propeller wash, which is blown directly over the wings from the propellers. This velocity, V_{prop} always moves along the x-axis, and is effectively added to the freestream velocity. V_{prop} can be calculated from conservation of momentum in fluid dynamics. The following equation was taught in Dr. Dave Peters' MEMS 5703 Analysis of Rotary-Wing Systems course, and gives a relationship between propeller thrust, freestream velocity, and V_{prop}:

$$T = \frac{1}{2}\rho\pi D^2 |V_\infty + V_{prop}| V_{prop} \tag{13}$$

With a known thrust and freestream velocity, the value of V_{prop} can be numerically calculated with this equation. Once this value is known, the new effective α is $\arctan(\frac{w}{u+V_{prop}})$ and V_∞ is $\sqrt{(u+V_{prop})^2 + v^2 + w^2}$ which is a result of adding the two velocity vectors. The corrected V_∞ is then used to calculate dynamic pressure $q_\infty = \frac{1}{2}\rho V_\infty^2$.

2.5.2 Lift and Drag

Lift is defined as $L = C_L(\alpha, f) q_\infty S_w$ where C_L is the coefficient of lift and is a function of angle of attack and the flap deflection. C_L can be broken up into components which are linear first-order approximations in the form of a partial derivative times an angle: $C_L = C_{L_0} + \frac{\partial C_L}{\partial \alpha}\alpha + \frac{\partial C_L}{\partial f} f$. These partial derivatives are conventionally written as C_{L_α} and C_{L_f}. The dependence on flap deflection is nominally linear within $\pm 20^\circ$ of flaps, so with servo limitations on flap deflection, this model will be accurate.

When α is outside the range of $\pm 10^\circ$, an aerodynamic stall occurs which rapidly decreases lift. Thus, $C_L(\alpha)$ can be approximated as linear between $\pm 10^\circ$ and zero outside this zone, with a steep dropoff back to zero as shown below:

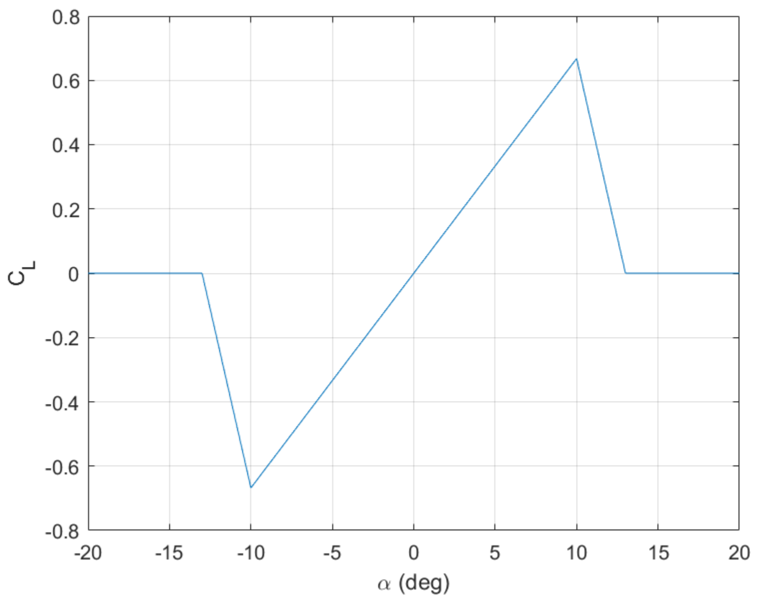

Figure 1: C_L vs. α

The value of the slope in the linear region can be determined from wind tunnel testing or with computational methods.

The drag equation is similar to lift: $D = C_D(\alpha, f)q_\infty S_w$ where $C_D = C_{D_0} + \frac{C_L^2}{\pi e AR}$. This is an approximation for lift-induced drag, where e is the Oswald efficiency factor and AR is the aspect ratio, which are both geometry dependent values. There also other methods of calculated drag as a function of angle of attack, which may be of interest in the future for a better model. This would most likely be by running a simple numerical solver (like AVL) at different values of α and using an interpolation lookup table.

Lift is always defined as being orthogonal to the freestream, and drag is parallel to the freestream (in the wind axes). If α is non-zero, then these forces need to be transformed into the body frame to be used in the equations of motion. Since the wind axes are just a rotation of $+\alpha$ about the y axis, the rotation matrix from wind to body frame is a negative rotation of α:

$$T_{W\to B} = \begin{bmatrix} \cos(\alpha) & 0 & -\sin(\alpha) \\ 0 & 1 & 0 \\ \sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix} \tag{14}$$

Then the body frame aerodynamic forces are:

$$F_A = T_{W\to B} \begin{bmatrix} -D \\ 0 \\ -L \end{bmatrix} = \begin{bmatrix} -D\cos(\alpha) + L\sin(\alpha) \\ 0 \\ -L\cos(\alpha) - D\sin(\alpha) \end{bmatrix} \tag{15}$$

It is important to note that L and D are the sum of the forces on both wings which could be different if the motor speeds or flap deflections are different.

In normal aircraft with a vertical surface like a tail with a rudder, there is a sideforce that is dependent on the sideslip angle β. Since the tailsitter has no vertical surfaces, β effects will be ignored.

2.5.3 Pitch and Roll Moments

Pitch moment has the equation $M_A = C_m(\alpha, f)q_\infty S_w \bar{c}$ where $\bar{c}$ is the mean aerodynamic chord. Like C_L with lift, C_m depends on α and has the same stall characteristics as the C_L curve since the pitch moment is a result of the lift distribution over the wings. Thus the equation for $C_m = C_{m_\alpha}\alpha + C_{m_f} f$ and these partial derivatives are determined with numerical methods like AVL or with empirical data. This is the only method of controlling the pitch rate of the tailsitter. Like lift and drag, the moment for each wing should be separately calculated and summed.

Roll moment is $L_{roll_A} = C_l(a) q_\infty S_w \bar{b}$ where $\bar{b}$ is the wingspan. C_l is the roll moment coefficient and only depends on the defined aileron flap deflection: $C_l = C_{l_a} a$. AVL defines +1 degree of aileron as +1 left flap and -1 right flap so this is how 1 degree of aileron will be defined for motor mixing. This means that $a = \frac{1}{2}(f_l - f_r)$, so then the roll from each wing can be calculated and summed, which takes into account the difference in dynamic pressure on each wing: $L_{roll} = \frac{1}{2}(L_{roll_l} - L_{roll_r})$.

Since there is no vertical surfaces, there is no aerodynamic yaw moment.

2.6 Dynamics Summary

Here is the full set of the nonlinear equations of motion:

$$\dot{\mathbf{V}} = -\boldsymbol{\omega} \times \mathbf{V} + \mathbf{T}_{E\to B} \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + \frac{1}{m} \begin{bmatrix} T - D\cos(\alpha) + L\sin(\alpha) \\ 0 \\ -L\cos(\alpha) - D\sin(\alpha) \end{bmatrix}$$

$$\dot{\mathbf{X}}_{NED} = (\mathbf{T}_{E\to B})^T \mathbf{V}$$

$$\dot{\boldsymbol{\omega}} = \mathbf{J}^{-1}[-\boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} + \begin{bmatrix} L_A + L_P \\ M_A \\ N_P \end{bmatrix}]$$

$$\dot{q} = \frac{1}{2} \begin{bmatrix} 0 & P & Q & R \\ -P & 0 & R & -Q \\ -Q & -R & 0 & P \\ -R & Q & -P & 0 \end{bmatrix} q$$

2.7 Simulink Plant Model

Below is the plant model in Simulink with the full nonlinear dynamics as described above. Rate transition blocks after the inputs simulate the zero-order hold of the input signals.

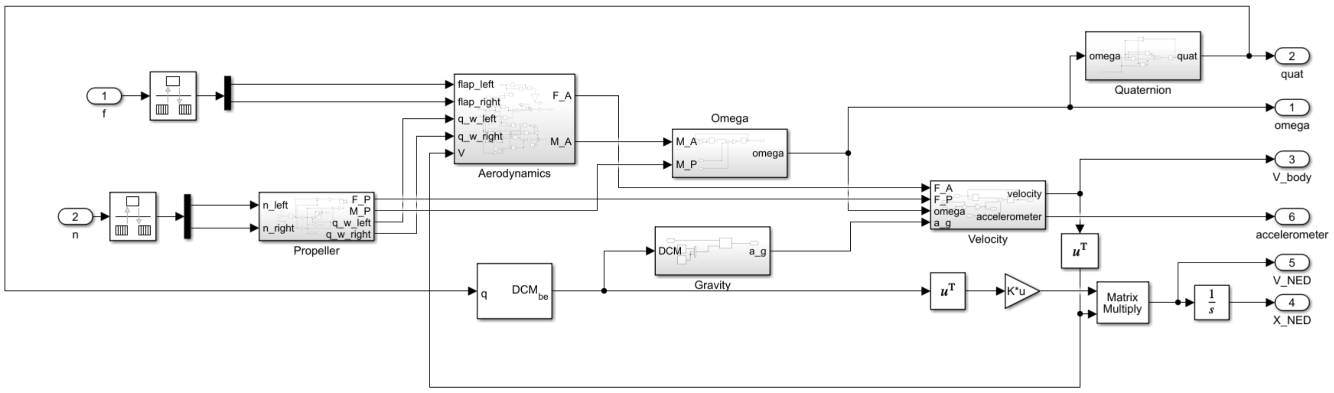

Figure 2: Tailsitter Plant

3 Control

3.1 Output Considerations

On the real tailsitter, there will be an accelerometer, a gyroscope, a barometer for altitude, a magnetometer for heading, and a GPS module. This means that not all states will be available for output. However, for the initial controller presented below to validate the model, all states were used as outputs. For real-life implementation, a Kalman filter will be needed to estimate attitude, velocity, and position.

3.2 Cascaded Control

Cascaded control is a controller architecture that has multiple feedback loops to stabilize different state variables like angular rate or position. The innermost loop is called the inner loop and typically involves angular rate stabilization. The angular velocities from the gyroscope are fed back and manipulated to form a control signal that will drive the angular velocity error to zero. The reference angular velocity is determined by the next outer loop, usually an attitude controller. The outer-most loop is usually a position or velocity controller that takes in a reference position/velocity and outputs a reference attitude, which then cascades into the attitude loop, and so on until the inner loop angular rate controller is reached. The fastest-acting loop should be the inner loop since its purpose is general stability, while the outer loops are meant for tracking a certain position or trajectory.

Cascaded control in this form is generally found in airplanes and missiles since they have decoupled longitudinal and lateral-directional dynamics in level flight, and each cascaded controller can be implemented independently. For the tailsitter in a hover, the vertical direction is an additional degree of freedom. This is directly controlled by varying the thrust and its purpose is to maintain a certain altitude and/or vertical velocity. Thus, for stability in this direction, the vertical acceleration must be zero. The inner loop will be split into a vertical acceleration loop and a standard angular velocity loop.

3.2.1 Motor Mixing

For cascaded control, there must be a connection between a desired motion and an input to the system. The standard four inputs for a conventional aircraft are Thrust, Elevator, Aileron, and Rudder (TEAR). Thrust is controlled by a collective propeller input, elevator is controlled by the pitching moment using collective flaps, aileron is controlled by the roll moment using differential flaps, and rudder is controlled by differential propeller inputs.

This means that the inputs can be split into TR (motors) and EA (flaps). One unit of T is +1 Hz on both motors. One unit of R is +1 Hz on the left motor and -1 Hz on the right motor. One unit of E is -1° on both flaps. One unit of A is +1° on the left flap and -1° on the right flap. In matrix form, these are:

$$\begin{bmatrix} n_l \\ n_r \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} T \\ R \end{bmatrix} \tag{16}$$

$$\begin{bmatrix} f_l \\ f_r \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ -1 & -1 \end{bmatrix} \begin{bmatrix} E \\ A \end{bmatrix} \tag{17}$$

These matrices allow for direct control of the four standard aircraft control inputs.

3.2.2 Actuator Models

A limit of ±20° was implemented on the flaps along with a rate limiter of 400 deg/s, which is a standard rate limit for hobby servos.

For the propellers, a saturation at 0 and 500 Hz was implemented, since the motor cannot spin backwards and will not spin faster than 500 Hz. Note that this limit is applied after the equilibrium hover speed is added to the control signal (for a hover).

A future plan could be to model the digital delays in these actuators, but this was not considered this semester and probably will not be necessary.

3.2.3 Block Diagrams

Below is a block diagram of an earlier Simulink model with an angular rate/acceleration inner loop and an attitude outer loop. In this diagram, the rate transition blocks at the outputs of the plant simulate analog-to-digital converters (ADC) by sampling the output signal at 200 Hz.

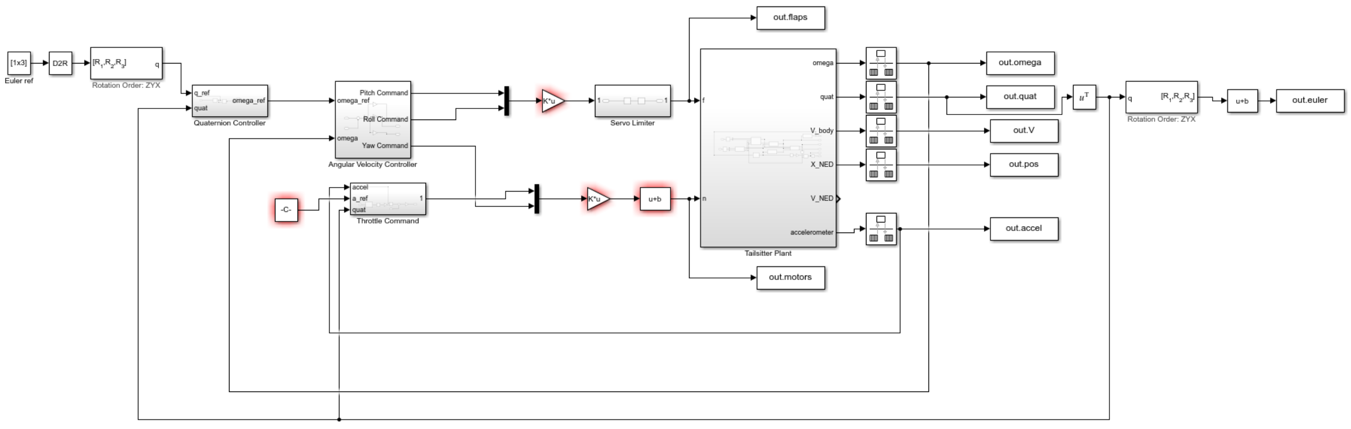

Figure 3: Cascaded Controller in Simulink

A simple block diagram model of the additional velocity outer loop is shown below, and is described in the following subsections.

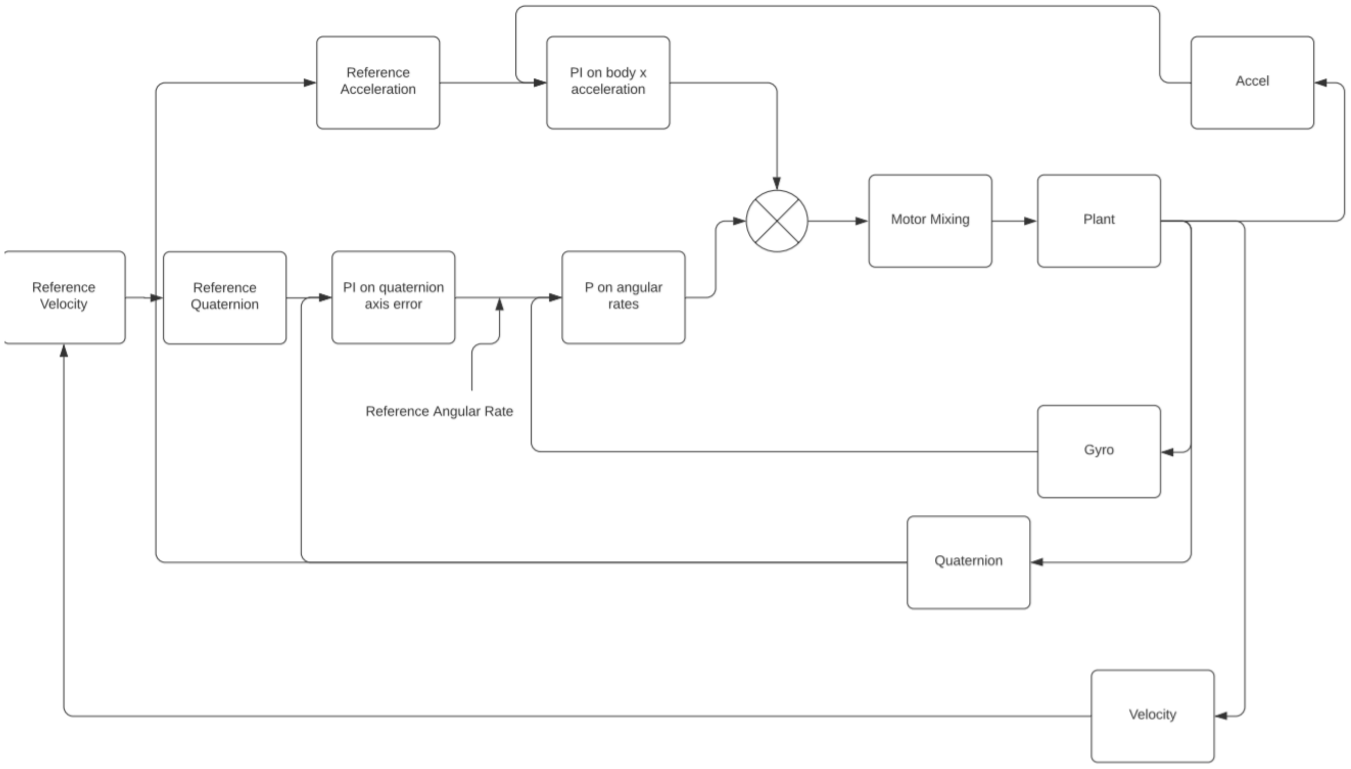

Figure 4: Cascaded Controller with Velocity Outer Loop

3.2.4 Inner Loop - Angular Velocity and Acceleration

The inner loop consists of angular velocity and vertical acceleration feedback. There are four connections for each of the four TEAR control inputs: P (roll rate) goes to Aileron, Q (pitch rate) goes to Elevator, R (yaw rate) goes to Rudder, and a_x (x-acceleration) goes to Thrust. Note that this acceleration feedback assumes that the tailsitter is in a near-hover.

These four connections are mostly decoupled, so they can operate in parallel without affecting the other control inputs. For example, a thrust or elevator command will not induce a roll or yaw. A yaw command will produce a slight roll as seen in the dynamics equations, but this effect is very minimal.

The acceleration reference is the opposite of gravity, so in this case $[0\ 0\ \ -g]^T$ which is rotated from the Earth frame to the body frame by the current quaternion state. For simplicity, PID controllers were used on the inner loop controller and tuned with a goal of driving the errors to zero within 0.5 seconds.

Below are the two controllers that make up the inner loop. The angular velocity controller shown was a P controller, but depending on the desired performance this can be a PID controller and was later changed to speed up the controller.

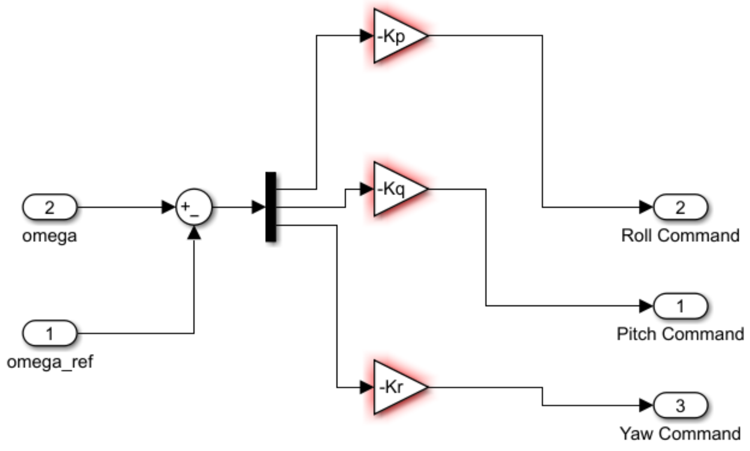

Figure 5: Angular Velocity Controller

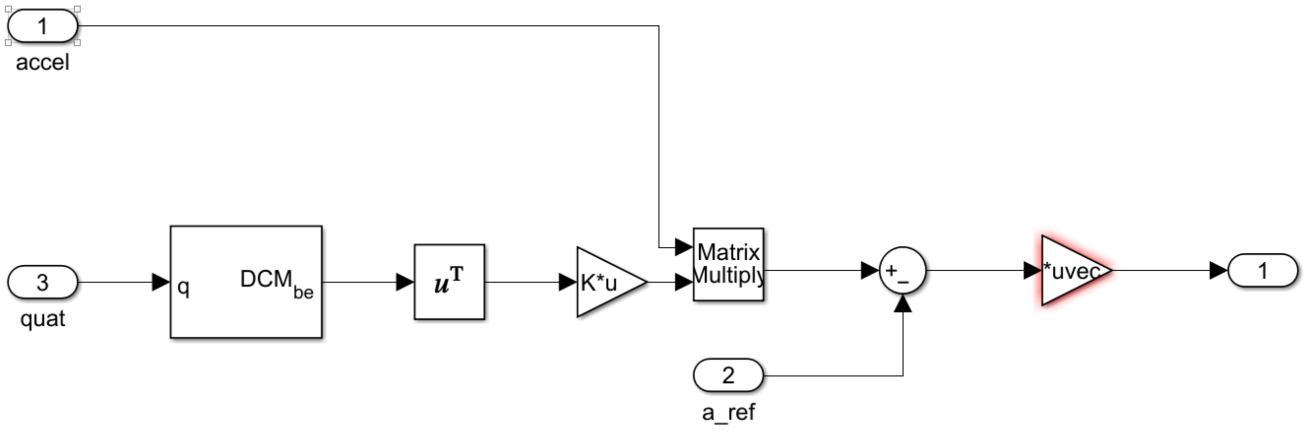

Figure 6: Vertical Acceleration Throttle Controller

3.2.5 Attitude Control Loop

The reference angular velocities for the inner loop are produced by the attitude control loop which uses a reference quaternion and quaternion feedback. In a realistic simulation of the available outputs, the quaternion state would have to be estimated with some type of observer. Assuming full state quaternion feedback, the following method is used to calculate the necessary control input to achieve the desired attitude [1]:

$$q_{err} = q_{ref} \otimes q* \tag{18}$$

where $\otimes$ represents quaternion multiplication and $q*$ is the conjugate of the quaternion feedback q. Then, take the imaginary components of q_{err} as a vector and multiply by the sign of the scalar part of q_{err}. This is now a 3x1 vector of errors that can be fed into a PID controller to produce the desired angular velocity to achieve the reference attitude. The axis error vector is in the order of roll, pitch, yaw (P, Q, R).

Below is a block diagram of the quaternion attitude controller with a simple proportional gain of -10, but this was also later tested with a PID controller.

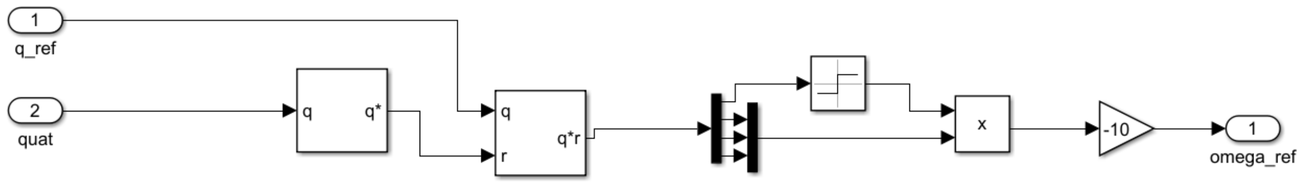

Figure 7: Quaternion Attitude Controller

3.2.6 Velocity Outer Loop

The velocity outer loop takes in reference velocities and outputs a reference attitude. The tilting of the tailsitter is how it can translate sideways, so a larger tilt angle will result in a higher velocity. This controller is for motion in the x-y plane, so it produces a reference quaternion that has reference pitch and yaw angles. The velocity outer loop uses a PID that is tuned to ensure it is the slowest of the three loops. It should be noted that the reference pitch and yaw angles are in the standard Euler angle format because they are easily matched to the corresponding velocity direction. For example, pitch angle always creates motion in the body x direction. Based on the method of estimating attitude, converting to quaternions may not be necessary, but this is a future consideration.

3.2.7 Position Outer Loop

A position outer loop was tested that was outside the velocity loop and produced a reference velocity. The inputs to the position controller were the x and y reference positions and the x and y position feedback. The outputs were the x and y reference velocities. The purpose of this loop was to command a specific position in the x-y plane by setting the necessary velocities to achieve that position. A PID controller was used to generate the reference velocities.

The problem with this controller was that the velocity would start out large as the position error was large which would cause some instabilities as the commanded angles were too large. In addition, this loop was the slowest and responded sluggishly to the errors in position. Depending on the tuning of the controller, the tailsitter position would enter a limit cycle and never achieve the desired position, or would come to a stop away from the setpoint, then move again to another location without coming to the setpoint. The tuning of the PID controller was very sensitive and small changes would result in instabilities, especially in the integral controller.

A suggested change to this controller would be a velocity-limited controller that flies the tailsitter at a constant velocity until it is close enough to the setpoint and then uses a traditional controller to reduce the error. This would solve some instability issues with a high reference velocity.

3.3 LQR Control [3]

Unlike a model using Euler angles for attitude, the 13 state quaternion model is not controllable because the quaternion uses four states to describe a 3 DOF rotation. This occurs because the scalar term in the quaternion is coupled with the three imaginary terms due to the constraint that $||q|| = 1$. This gives a formula for q_0 in terms of the other components: $q_0 = \sqrt{1 - q_1^2 - q_2^2 - q_3^2}$.

An equivalent quaternion differential equation that splits the scalar term from the vector terms is below:

$$\dot{\boldsymbol{q}} = \frac{1}{2} q_0 \boldsymbol{\omega} - \frac{1}{2} \boldsymbol{\omega} \times \boldsymbol{q} \tag{19}$$

$$\dot{q_0} = -\frac{1}{2}\boldsymbol{\omega^T q} \tag{20}$$

where $\boldsymbol{q}$ is the vector part of q, or $[q_1\ q_2\ q3]^T$.

The formula is then used in the quaternion differential equation in place of q_0, and the equation becomes a 3x1 system from a 4x1 system:

$$\dot{\boldsymbol{q}} = \frac{1}{2}\sqrt{1-\boldsymbol{q^T q}}\boldsymbol{\omega} - \frac{1}{2}\boldsymbol{\omega} \times \boldsymbol{q} \tag{21}$$

Using this new equation, the whole system is now controllable and an LQR controller can be made.

Linearization was completed using a symbolic Jacobian in MATLAB and plugging in the hover equilibrium states and inputs, which are all zero except for the quaternion state, the altitude, and the motor input.

A Q matrix using the following diagonal entries was used: $diag([0\ 0\ 0\ 100\ 100\ 100\ 0\ 0\ 0\ 1\ 1\ 100])$ along with a 4x4 identity matrix for R. This Q matrix puts no penalties on the angular rates and translational velocities, but highly penalizes changes in attitude and altitude.

4 Future Ideas and Next Steps

- Test 2D longitudinal (pitch only) model in the learning method simulation
- Fly the tailsitter using the cascaded control system currently running in Simulink

References

[1] Jonathan P How, Emilio Frazzoli, and Girish Vinayak Chowdhary. "Linear flight control techniques for unmanned aerial vehicles". In: *Handbook of unmanned aerial vehicles.* Springer Netherlands, 2015, pp. 529–576.

[2] Brian Stevens. *Aircraft Control and Simulation.* 2016.

[3] Yaguang Yang. "Quaternion-Based LQR Spacecraft Control Design Is a Robust Pole Assignment Design". In: *Journal of Aerospace Engineering* 27.1 (2014), pp. 168–176. DOI: `10.1061/(ASCE)AS.1943-5525.0000232`.