Tailsitter Equations of Motion

Daniel Cherenson

Updated October 1 2020

1 Introduction

This document describes the build up of the equations of motion for the Tailsitter. I will start with deriving the equations for a hover for use in a MATLAB simulation.

The vehicle 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.

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

The inputs to the tailsitter are flap deflections δ_l and δ_r and two propeller speeds n_l and n_r.

We will use quaternions to describe the orientation of the tailsitter. The alternative method is to use Euler angles which are more physically intuitive, but at certain values, there is a singularity in their differential equation. This singularity is usually at a pitch angle of $\pm\frac{\pi}{2}$ depending on the definition of the Euler angles, and this will be the orientation of the tailsitter in hover. Using quaternions avoids this singularity and is easy to implement mathematically and computationally in MATLAB.

2 Hover Equations of Motion

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: $q = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T$
- Angular velocity vector: $\boldsymbol{\omega} = \begin{bmatrix} P & Q & R \end{bmatrix}^T$

2.1 Velocity Differential Equation

This will give us a differential equation in the form $\dot{\mathbf{V}} = f(\mathbf{v}, \mathbf{q}, \boldsymbol{\omega}, \mathbf{F})$ 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 us how to convert this to the vehicle frame (subscript V), given the following equation which assumes constant mass:

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

We do this because the accelerometer measures accelerations in the vehicle 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. The equation for this matrix is below [1]:

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

Plugging in $\mathbf{F}_G = \mathbf{T}_{E\to B}(q) \begin{bmatrix} 0 & 0 & mg \end{bmatrix}^T$ and simplifying the velocity equation, we get:

$$\dot{\mathbf{V}} = -\boldsymbol{\omega} \times \mathbf{V} + g \begin{bmatrix} 2(q_1q_3+q_0q_2) \\ 2(q_2q_3+q_0q_1) \\ (q_0^2-q_1^2-q_2^2+q_3^2) \end{bmatrix} + \frac{1}{m}(\mathbf{F}_A + \mathbf{F}_P) \tag{4}$$

Looking at the aerodynamic and propulsive forces, we can remove some terms. For hover we will assume angle of attack is 0 since there is no relative wind, which means that lift will always act in the negative z direction. The only force aerodynamic force acting on the tailsitter is the lift from airflow over the wings when the flaps are deflected. We assume drag is negligible because the tailsitter will have a very low velocity in hover.

With these assumptions, $\mathbf{F}_A = \begin{bmatrix} 0 & 0 & -L(\delta, n) \end{bmatrix}^T$ which sums the lift components from each wing. Likewise, the propulsive force only acts in the positive x direction, so $\mathbf{F}_P = \begin{bmatrix} T(n) & 0 & 0 \end{bmatrix}^T$

Finally, the differential equation for velocity in terms of the states and inputs is:

$$\dot{\mathbf{V}} = -\boldsymbol{\omega} \times \mathbf{V} + g \begin{bmatrix} 2(q_1 q_3 + q_0 q_2) \\ 2(q_2 q_3 + q_0 q_1) \\ (q_0^2 - q_1^2 - q_2^2 + q_3^2) \end{bmatrix} + \frac{1}{m} \begin{bmatrix} T(n) \\ 0 \\ -L(\delta, n) \end{bmatrix} \tag{5}$$

2.2 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:

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

2.3 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 vehicle 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 $J_x y$ and $J_x z$ 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:

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

The aerodynamic moments are roll and pitch moments. Opposite deflection of the flaps causes a roll and a coordinated deflection of the flaps causes a pitch. There is no vertical surface so there is no yaw moment from aerodynamic forces. The aerodynamic moment vector is $\mathbf{M}_A = \begin{bmatrix} L_A(\delta, n) & M_A(\delta, n) & 0 \end{bmatrix}^T$. This L should not be confused with lift - it is the roll moment (sorry, this is just aerodynamics convention).

For the propulsive moments, a differential thrust will cause a yaw. Because the propellers are counter-rotating, when they are at the same speed, they will balance each other out. However, when one rotates at a different speed, conservation of angular momentum will cause the tailsitter to rotate, or roll. We may be able to neglect this propulsive roll moment L_P since the differences in motor speeds will probably be small. For now, we will write the propulsive moments as $\mathbf{M}_P = \begin{bmatrix} L_P(n) & 0 & N_P(n) \end{bmatrix}^T$.

Finally, the angular velocity differential equation is:

$$\dot{\boldsymbol{\omega}} = \mathbf{J}^{-1}[-\boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} + \begin{bmatrix} L_A(\delta, n) + L_T(n) \\ M_A(\delta, n) \\ N_P(n) \end{bmatrix}] \tag{8}$$

2.4 Quaternion Differential Equation

Finally, we have the differential equation for the orientation state. From [1], the quaternion differential equation is a function of q and $\boldsymbol{\omega}$, and is given as:

$$\dot{q} = \frac{1}{2} \begin{bmatrix} 0 & -P & -Q & -R \\ P & 0 & R & -Q \\ Q & -R & 0 & P \\ R & Q & -P & 0 \end{bmatrix} q \tag{9}$$

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 [1]:

$$q_0 = \begin{bmatrix} cos\frac{\phi}{2}cos\frac{\theta}{2}cos\frac{\psi}{2} + sin\frac{\phi}{2}sin\frac{\theta}{2}sin\frac{\psi}{2} \\ sin\frac{\phi}{2}cos\frac{\theta}{2}cos\frac{\psi}{2} - cos\frac{\phi}{2}sin\frac{\theta}{2}sin\frac{\psi}{2} \\ cos\frac{\phi}{2}sin\frac{\theta}{2}cos\frac{\psi}{2} + sin\frac{\phi}{2}cos\frac{\theta}{2}sin\frac{\psi}{2} \\ cos\frac{\phi}{2}cos\frac{\theta}{2}sin\frac{\psi}{2} - sin\frac{\phi}{2}sin\frac{\theta}{2}cos\frac{\psi}{2} \end{bmatrix} \tag{10}$$

3 Summary

Here is the full set of the nonlinear equations of motion which describe the tailsitter in a hover:

$$\dot{\mathbf{V}} = -\boldsymbol{\omega} \times \mathbf{V} + g \begin{bmatrix} 2(q_1 q_3 + q_0 q_2) \\ 2(q_2 q_3 + q_0 q_1) \\ (q_0^2 - q_1^2 - q_2^2 + q_3^2) \end{bmatrix} + \frac{1}{m} \begin{bmatrix} T(n) \\ 0 \\ -L(\delta, n) \end{bmatrix}$$

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

$$\dot{\boldsymbol{\omega}} = \mathbf{J}^{-1}[-\boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} + \begin{bmatrix} L_A(\delta, n) + L_T(n) \\ M_A(\delta, n) \\ N_P(n) \end{bmatrix}]$$

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

4 Next Steps

The next steps are to make models for the aerodynamic and propulsive forces and moments as functions of the inputs δ and n, and them make the simulation in MATLAB.

References

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