# Rocket Ramblings

Pazuki Development Notes

Het Satasiya

November 22, 2024

# Chapter 1

# Rocket Math

## 1.1 Introduction

This is a collection of my ramblings on how to make a rocket controlled, and hopefully land. Its going to start with the base knowledge that you need that wont be covered in a course like 9A, and an understanding of how to do basic matrix algebra ($A \times \vec{x}$, $A \times B$, $\vec{x} \cdot \vec{y}$). You also need to know how to take a derivative.

To take a rocket and control it, you first need to understand what makes it move / makes the system tick. This gives you a basis to create simulations off of, and you can linearize the equations of motion to find your system dynamics in a linearized-state space model. You can use a variety of control algorithms, varying from Bang-Bang control, PID, LQR based control, Cascade control, or Model Predictive Control (MPC). Lets get cooking.

## 1.2 Basis

### 1.2.1 Reference Frames

A reference frame is what you measure everything relative to. For example, majority of physis problems are in the 'Earth-Fixed' or 'Inertial' reference frame. What that means is that you have a stationary observer, and all measurements are 'relative' to that stationary point. But if you were in the perspective of a moving object, then your observing a 'relative' velocity and position. This creates the necessity for an equation known as the 'equation of coriolis' or 'something rule' depending on what dynamics professor you have. You need to use this to convert from one reference frame to another, as you have to consider the rate of angular movement around each other.

$$\frac{\mathrm{d}}{\mathrm{d}t_i}\vec{p} = \frac{\mathrm{d}}{\mathrm{d}t_p}\vec{p} + \omega_{p/i} \times \vec{p} \tag{1.1}$$

I wont go into the derivation, but its written above in equation 1.1. You can use it to convert the derivative of some position or velocity vector from one reference frame $p$ to another reference frame $i$.

### 1.2.2 Rotation

How you look at rotation is very important, especially when you need to maintain some form of consistency. You can represent rotation in a variety of ways, but the two that we will be considering are Euler angles, and Quaternions. Before diving into that however, your end goal needs to be to describe a rotation and rotational rate between two reference frames. The reason you cant describe rotation with just one reference frame is because of what a rotation is. If your spinning your phone, you define a 'Forward', 'Up', and 'Side' direction to reference a rotation around. In the same way, you must mathematically define a rotation as a transformation between two reference frames. Even though a reference frame is technically 'fixed' to a point (like the earth fixed frame), you can 'move' (in a sense) the frames to have their origins lie on top of each other to understand the rotations required to take you from one to the next. (This is the underlying

principle behind Euler angles). Quaternions define rotation as a 4 dimensional vector. For further info, look here.

#### Euler Angle Rotations

So, onto some math for rotation. Euler angle rotations can be divided up into 3 different matrices / principle rotations.

To rotate a vector around the $x$ axis:

$$\vec{x}_f = R_x(\phi) \times \vec{x}$$
$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{1.2}$$

To rotate a vector around the $y$ axis:

$$\vec{x}_f = R_y(\psi) \times \vec{x}$$
$$R_y(\psi) = \begin{bmatrix} \cos(\psi) & 0 & \sin(\psi) \\ 0 & 1 & 0 \\ -\sin(\psi) & 0 & \cos(\psi) \end{bmatrix} \tag{1.3}$$

To rotate a vector around the $y$ axis:

$$\vec{x}_f = R_z(\theta) \times \vec{x}$$
$$R_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.4}$$

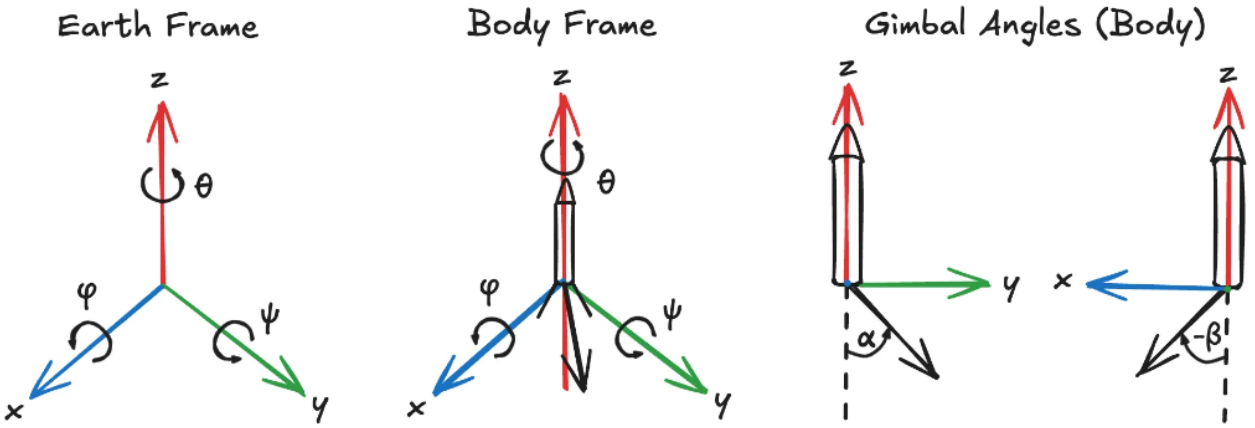
And to sum it up, to rotate a vector around all axes in order, you multiply the matrices from 1.2, 1.3, 1.4, in that order, giving us:

$$R_E^B = \begin{bmatrix} c\psi c\theta & c\psi s\theta & -s\psi \\ s\phi s\psi c\theta - c\phi s\theta & s\phi s\psi s\theta + c\phi c\theta & s\phi c\psi \\ c\phi s\psi c\theta + s\phi s\theta & c\phi s\psi s\theta - s\phi c\theta & c\phi c\psi \end{bmatrix} \tag{1.5}$$

### 1.2.3 Variable names

We have two reference frames, a body fixed reference frame $B$, and a earth fixed (inertial) reference frame $E$.

Figure 1.1: Coordinate Systems, and positive notation



Further notation / notes: $[x,\ y,\ z]$ are earth frame coordinates / positions. $[u,\ v,\ w]$ are body frame velocities in the $x$, $y$, and $z$ axes, respectively. Angular rotation is represented by $[\phi,\ \psi,\ \theta]$, with angular rates as $[p,\ q,\ r]$. Gimbal angles are represented by $\alpha$ and $\beta$, representing right hand rotations around the $x$ and $y$ axes respectively. The rotation order that we will be using is X-Y-Z.

## 1.3 Dynamics

At the heart of any controller, you need to have a strong understanding of what your system dynamics are. Your system dynamics can also be thought of as the 'governing equations' that determine the motion of your craft in response to external forces.

We can define your control inputs as a force vector$\vec{F} = (f_x, f_y, f_z)$, and your control torques as $\vec{m} = (\tau_\phi, \tau_\psi, \tau_\theta)$.

### 1.3.1 Rocket Kinematics

In this section we are going to find the equations that determine position wrt velocity, and rotation wrt angular rates.

$x, y, z$ are all inertial frame quantities, so you have to rotate the body frame velocities $u, v, w$. You can do that by transposing the $R_B^E$ matrix we found earlier in equation 1.5.

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} x \\ y \\ z \end{bmatrix} &= R_B^E \begin{bmatrix} u \\ v \\ w \end{bmatrix} \\ &= (R_E^B)^T \begin{bmatrix} u \\ v \\ w \end{bmatrix} \end{aligned} \tag{1.6}$$

Rotational dynamics are harder, because you have to account for intermediate rotations. Also, by and large, your rotational velocities are pretty low, so your $R_x(\dot{\phi}) = R_y(\dot{\psi}) = R_x(\dot{\theta}) = I$.

$$\begin{aligned} \begin{bmatrix} p \\ q \\ r \end{bmatrix} &= R_x(\dot{\phi}) \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + R_x(\phi)R_y(\dot{\psi}) \begin{bmatrix} 0 \\ \dot{\psi} \\ 0 \end{bmatrix} + R_x(\phi)R_y(\psi)R_z(\dot{\theta}) \begin{bmatrix} 0 \\ 0 \\ \dot{\theta} \end{bmatrix} \\ &= \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + R_x(\phi) \begin{bmatrix} 0 \\ \dot{\psi} \\ 0 \end{bmatrix} + R_x(\phi)R_y(\psi) \begin{bmatrix} 0 \\ 0 \\ \dot{\theta} \end{bmatrix} \end{aligned} \tag{1.7}$$

### 1.3.2 Rigid Body Dynamics

Forces and their respective accelerations are in the inertial fixed frame, so we can use the equation of coriolis (1.1). $\vec{v}$ is the velocity of the object.

$$\begin{aligned} m\frac{\mathrm{d}}{\mathrm{d}t_E}\vec{v}_E = m\left(\frac{\mathrm{d}}{\mathrm{d}t_B}\vec{v}_B + \vec{\omega}_{B/E} \times \vec{v}\right) &= \vec{F} \\ \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} &= \begin{bmatrix} rv - qw \\ pw - ru \\ qu - pv \end{bmatrix} + \frac{1}{m}\begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \end{aligned} \tag{1.8}$$

Moments of inertia are written as $J_x$, $J_y$, $J_z$, along those axes (shortened from $J_{xx}, J_{yy}, J_{zz}$). If you ignore the cross products of inertia ($J_{xy}, J_{xz}$, etc), it makes your life a lot easier. And if you define your control input torques $\vec{m} = (\tau_\phi, \tau_\psi, \tau_\theta)$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} \frac{J_y - J_z}{J_x} qr \\ \frac{J_z - J_x}{J_y} pr \\ \frac{J_x - J_y}{J_z} pq \end{bmatrix} + \begin{bmatrix} \frac{1}{J_x} \tau_\phi \\ \frac{1}{J_y} \tau_\psi \\ \frac{1}{J_z} \tau_\theta \end{bmatrix} \tag{1.9}$$

And thats a wrap for our dynamics 🙂.

# Chapter 2

# Control

Heres the fun part, which is trying to control your system. There are so many ways to do it, but lets start with some more important definitions. The thing that you want to control is called your 'system' or 'plant'. Whatever variables that you can apply to modify your system are your 'control inputs'. And the different quantities of your system that you can measure and want to control are called your 'states'.

## 2.1 Control Algorithms

Now that we have that out of the way, lets define a little toy system to learn about everything here. Lets start with our equations of motion.

We can define our states as our vertical position $y$ and our vertical velocity $v$. Our control input is a force in the y-axis, $F$. The object has a mass $m = 1kg$, and an acceleration from gravity $g = 9.81m/s^2$ on it.

$$\vec{x} = \begin{bmatrix} y \\ v \end{bmatrix}, \dot{\vec{x}} = \begin{bmatrix} \dot{y} \\ \dot{v} \end{bmatrix}, \vec{u} = \begin{bmatrix} F \end{bmatrix} \tag{2.1}$$

$$\begin{aligned} y &= vt \\ v &= \left(\frac{F}{m}\right) t - g * t \end{aligned} \tag{2.2}$$

That should be enough to make a basic simulation of our object using some python or MATLAB and Simulink depending on what you want to do.

### 2.1.1 PID Control

Lets create a simple controller, and explore what driving a system to a state looks like and how modifying relatively simple variables changes the response time.