About Quaternions and Rotations

Basics

Quaternions

Basics

Imaginary numbers $i = \sqrt{-1}$ or $i^2 = -1$

$a + bi = r(\cos\theta + i\sin\theta) = re^{i\theta}$

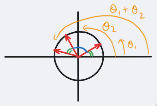
{ Euler's Formula : $e^{i\theta} = \cos\theta + i\sin\theta$ } → polar form

Fun Fact: $\Rightarrow e^{i\pi} + 1 = 0$

A direct consequence of Euler's Formula :

$z_1 = e^{i\theta_1}$, $z_2 = e^{i\theta_2}$ } → 2 unit complex numbers

$z_1 \cdot z_2 = e^{i\theta_1} \cdot e^{i\theta_2} = e^{i\theta_1 + i\theta_2} = e^{i(\theta_1 + \theta_2)}$



" Multiplying complex numbers ||| Rotation and Scaling in Complex Plain "

Quaternions are the mathematical numbers that describe rotations in 3-Dimensions in the same way Complex numbers define rotations in 2-Dimensions

Instead of 3 numbers, it turns out you actually need 4 nubers to completely describe 3-D rotations

Complex # : $a + bi$ (a: Real, bi: imaginary)

Quaternions : $a + bi + cj + dk$ (a: Scalar Part, bi + cj + dk: Vector Part)

For a quaternion $q = a + bi + cj + dk$

$|q| = \sqrt{a^2 + b^2 + c^2 + d^2}$

Just the way unit complex numbers ($|z| = 1$, $z \in \mathbb{C}$)

For Quaternions, an additional constraint is :

$i^2 = j^2 = k^2 = ijk = -1$

$a^2 + b^2 = 1 \Rightarrow$ circle { curve in 2-D space }

$a^2 + b^2 + c^2 = 1 \Rightarrow$ sphere { Surface in 3-D space }

$a^2 + b^2 + c^2 + d^2 = 1 \Rightarrow$ 4-D Hypersphere { Volume in 4-D space }

Unit Quaternions (sort of) define 3-dimensional Rotations (With a bijection)

> Quaternion Vs Rotation Matrix
>
> Every (2) Quaternion defines a unique rotational state and every rotational state can be translated to a unique quaternion
>
> Matrices are not a good due to the lack of this bijection

> Quaternions are Computationally more efficient that Rotation Matrices { 4 numbers vs 9 }

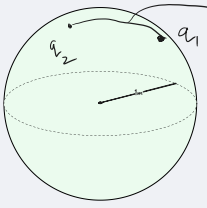
> In principal, can be internally translated { In fact many game engine routines use quaternions as a black box for processing, under the hood }

> Quaternions don't have problems like gimbal locking, etc

unit quaternion q_1, q_2 live in a 4-D Hypersphere $(a^2 + b^2 + c^2 + d^2 = 1)$

a slice of this with angle $\theta = \frac{\pi}{2}$ { $\cos\theta = 0$, $\sin\theta = 1$ } would be

$0 + b^2 + c^2 + d^2 = 1$



→ Geodesic { shortest path for the given constraint surface }

In case the angle is not changing, your geodisic { $slerp(q_1, q_2, t)$ } is a great circle in the 3-D surface.

But for rotations + change of axis you don't need 2 rotations

instead, $slerp(q_1, q_2, t)$ gives you a single, most efficient rotation that is a the shortest path in the 4-D Hyperersphere, but not neccessarily in the 3-D Sphere we see.

https://en.wikipedia.org/wiki/Euler%27s_rotation_theorem#:~:text=In%20geometry%2C%20Euler's%20rotation%20theorem,runs%20through%20the%20fixed%20point.

Usage

Quaternions

Usage

So How do you use a quaternion to describe rotations?

$$q = a + bi + cj + dk$$

if the quaternion is a unit quaternion,

$$a^2 + b^2 + c^2 + d^2 = 1, \quad i^2 = j^2 = k^2 = ijk = -1$$

it turns out we can thus write Euler's Formula for Quaternions

$$q = e^{\frac{\theta}{2}\hat{n}} = \cos\left(\frac{\theta}{2}\right) + \hat{n}\sin\left(\frac{\theta}{2}\right)$$

where $\hat{n}$ = unit pure quaternion

$$\hat{n} = x\hat{i} + y\hat{j} + z\hat{k} = \frac{b\hat{i} + c\hat{j} + d\hat{k}}{\sqrt{b^2 + c^2 + d^2}}$$

$$q = a + b\hat{i} + c\hat{j} + d\hat{k} = \cos\left(\frac{\theta}{2}\right) + \hat{n}\sin\left(\frac{\theta}{2}\right)$$

$\hat{n}$ ↓ norm (vector part (Quaternion))

then, quite conveniently, it turns out that if you have a vector $v = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$. if you write that as a pure quaternion

$$q_v = 0 + v_1\hat{i} + v_2\hat{j} + v_3\hat{k}$$

then,

$q\, q_v\, q^{-1}$ would give a quaternion whose vector part would be v rotated by angle θ around the axis $\hat{n}$ for

$$q = \cos\left(\frac{\theta}{2}\right) + \hat{n}\sin\left(\frac{\theta}{2}\right)$$

$$\{\hat{n} = x\hat{i} + y\hat{j} + z\hat{k}\}$$

Furthermore, this transformation is true not only for vectors but rotational quaternions themselves.

So if you have a rotation quaternion, say q_1 that rotates a vector 90° about x axis

So q_1 = axis angle 2 quat $\left([1,0,0], \frac{\pi}{2}\right) = \cos\left(\frac{\pi/2}{2}\right) + (\hat{i} + 0 + 0)\sin\left(\frac{\pi/2}{2}\right)$

$$= \frac{1}{\sqrt{2}} + \frac{i}{\sqrt{2}}$$

and I want to transform this rotation itself such that it becomes a 90° rotation around about the local coordinates, upon transformation q_2, say a 90° rotation about z-axis

q_2 = axis angle 2 quat $\left([0,0,1], \pi/2\right) = \frac{1}{\sqrt{2}} + \frac{\hat{j}}{\sqrt{2}}$

you essentially sandwich q_1 between q_2 & q_2^{-1}

$$\Rightarrow q_{eq} = q_2\, q_1\, q_2^{-1}$$

{ If $q_2 = \cos\left(\frac{\theta}{2}\right) + \hat{n}\sin\left(\frac{\theta}{2}\right)$ rotates by θ angle about $\hat{n}$,

$q_2^{-1} = \cos\left(-\frac{\theta}{2}\right) + \hat{n}\sin\left(-\frac{\theta}{2}\right)$ would rotate by $-\theta$ angle about $\hat{n}$, hence $\text{inv}(q) = \text{scalar}(q) - \text{vector}(q)$ {for unit quaternion q} }

$$\Rightarrow q_{eq} = \left(\frac{1}{\sqrt{2}} + \frac{\hat{k}}{\sqrt{2}}\right)\left(\frac{1}{\sqrt{2}} + \frac{\hat{i}}{\sqrt{2}}\right)\left(\frac{1}{\sqrt{2}} - \frac{\hat{k}}{\sqrt{2}}\right)$$

$$= \frac{1}{\sqrt{2}} + \frac{\hat{j}}{\sqrt{2}} \quad \{\hat{k}\,\hat{i}\,\hat{j}\}$$

= 90° rotation about y-axis in global space.

In general, to transform a quaternion

form space$_1$ to space$_2$ you do the following:

say you rotate / transform a quaternion q_1 defined as $q_{1s} + q_{1x}\hat{i} + q_{1y}\hat{j} + q_{1z}\hat{k}$ by quaternion Q_1 :

$$q_0 = Q_1 \, q_1 \, Q_1^{-1}$$

to see what q_1 in Q_1 space would be equivalent to for another state Q_2, we want q_2 : $q_0 = Q_2 \, q_2 \, Q_2^{-1}$

i.e. we transforming out of states Q_1 & Q_2 by the sandwich multiplication, both $Q_1 q_1 Q_1^{-1}$ & $Q_2 q_2 Q_2^2$ map to the same physical vector in std. Bases.

$\Rightarrow$ since $q_0 = Q_2 \, q_2 \, Q_2^{-1}$

$$q_2 = Q_2^{-1}\left(Q_2 \, q_2 \, Q_2^{-1}\right) Q_2 = Q_2^{-1}(q_0) Q_2$$

$$\Rightarrow q_2 = Q_2^{-1}(q_0) Q_2 \quad ** \rightarrow$$

so $Q(q)Q^{-1}$ gets you from Q state to std state & $Q^{-1}(q)Q$ gets you from std state to Q state.

but $q_0 = Q_1 \, q_1 \, Q_1^{-1}$

$$\Rightarrow q_2 = Q_2^{-1}\left(Q_1 \, q_1 \, Q_1^{-1}\right) Q_2$$

$$= \left(Q_2^{-1} Q_1\right) q_1 \left(Q_1^{-1} Q_2\right)$$

$$\boxed{q_2 = \left(Q_2^{-1} Q_1\right) q_1 \left(Q_2^{-1} Q_1\right)^{-1}} \quad \left\{(Q_1 Q_2)^{-1} = Q_2^{-1} Q_1^{-1} \;\&\; (Q_1^{-1})^{-1} = Q_1\right\}$$

$\therefore$ $Q^* = Q_2^{-1} Q_1$ would be the change of such that any rotation/vector q in defined in Q_1 state would have the SAME PHYSICAL REPRESENTATION in the std. Basis as when $Q^* \, q \, Q^{*-1}$ is mapped from Q_2 to the std. Basis.

this is the inner working of Conversions.Coordinate_transform method.

Rotation Matrices and Euler Angles

Quaternions

Rotation Matrices & Euler Angles

I would say one of the best resources on the topic would be Orientation, Rotation, Velocity and Acceleration, and the SRM https://www.sedris.org/wg8home/Documents/WG80485.pdf

It is important to understand the distinction b/w different conversions, ZYZ, ZXZ, etc ... Right-handed Coordinate systems vs Left-handed coordinate systems & Most importantly, the space of each operation – Local vs Global

Table 4 – Principal factors for *z-y-x* rotation

Case	Principal factors for rotation $R_z(\psi)R_y(\theta)R_x(\phi)$ (all angles modulo 2π)		
$a_{31} \neq \pm 1$	$\theta = \arcsin(-a_{31})$ [principal value] $-\pi/2 < \theta < \pi/2$	$\phi = \arctan2(a_{32}, a_{33})$	$\psi = \arctan2(a_{21}, a_{11})$
	$\theta = \arcsin(-a_{31})$ [π – principal value] $\pi/2 < \theta < 3\pi/2$	$\phi = \arctan2(-a_{32}, -a_{33})$	$\psi = \arctan2(-a_{21}, -a_{11})$
$a_{31} = -1$	$\theta = \pi/2$	$\phi = \arctan2(a_{12}, a_{13}) + \psi$	any value of ψ
$a_{31} = +1$	$\theta = -\pi/2$	$\phi = \arctan2(-a_{12}, -a_{13}) - \psi$	any value of ψ

Observe that order of the three rotation angles is reversed between the space-fixed and body-fixed cases:

$R_z(\gamma)R_x(\beta)R_z(\alpha)$ space-fixed

$$\boldsymbol{R}_{z''}(\alpha)\boldsymbol{R}_{x'}(\beta)\boldsymbol{R}_{z}(\gamma) \quad \text{body-fixed} \tag{1.6}$$

To show that both expressions produce the same rotation, note that when x' is at its intermediate position on the line of nodes, the second rotation $\boldsymbol{R}_{x'}(\beta)$ is equivalent to first rotating the line of nodes to the x-axis using principal rotation $\boldsymbol{R}_{z}(-\gamma)$, rotating about the x-axis (which is the line of nodes at this point) with $\boldsymbol{R}_{x}(\beta)$ and finally rotating the line of nodes back to its original position with $\boldsymbol{R}_{z}(\gamma)$. In effect, $\boldsymbol{R}_{x'}(\beta)=\boldsymbol{R}_{z}(\gamma)\boldsymbol{R}_{x}(\beta)\boldsymbol{R}_{z}(-\gamma)$. Similarly, $\boldsymbol{R}_{z''}(\alpha)=\boldsymbol{R}_{x'}(\beta)\boldsymbol{R}_{z}(\alpha)\boldsymbol{R}_{x'}(-\beta)$. Noting that two rotations about the same axis commute and substituting these expressions in the body-fixed formulation gives:

$$\begin{aligned}
\boldsymbol{R}_{z''}(\alpha)\boldsymbol{R}_{x'}(\beta)\boldsymbol{R}_{z}(\gamma) &= \left[\boldsymbol{R}_{x'}(\beta)\boldsymbol{R}_{z}(\alpha)\boldsymbol{R}_{x'}(-\beta)\right]\boldsymbol{R}_{x'}(\beta)\boldsymbol{R}_{z}(\gamma)\\
&= \left[\boldsymbol{R}_{x'}(\beta)\boldsymbol{R}_{z}(\alpha)\right]\boldsymbol{R}_{z}(\gamma)\\
&= \left[\left\{\boldsymbol{R}_{z}(\gamma)\boldsymbol{R}_{x}(\beta)\boldsymbol{R}_{z}(-\gamma)\right\}\boldsymbol{R}_{z}(\alpha)\right]\boldsymbol{R}_{z}(\gamma)\\
&= \boldsymbol{R}_{z}(\gamma)\boldsymbol{R}_{x}(\beta)\boldsymbol{R}_{z}(\alpha)\boldsymbol{R}_{z}(-\gamma)\boldsymbol{R}_{z}(\gamma)\\
&= \boldsymbol{R}_{z}(\gamma)\boldsymbol{R}_{x}(\beta)\boldsymbol{R}_{z}(\alpha)
\end{aligned}$$

This result:

$$\boldsymbol{R}_{z''}(\alpha)\boldsymbol{R}_{x'}(\beta)\boldsymbol{R}_{z}(\gamma)=\boldsymbol{R}_{z}(\gamma)\boldsymbol{R}_{x}(\beta)\boldsymbol{R}_{z}(\alpha) \tag{1.7}$$

shows that the space-fixed and body-fixed formulations produce the same rotation. Both formulations are important. The matrix formulations of the principal rotations (Equation (1.15)) are expressed with respect to the static space-fixed frame. However, an inertial system attached to the body would read out the angles with respect to the rotating body-fixed frame.

The orientation of the $\tilde{x},\tilde{y},\tilde{z}$ axes with respect to the x,y,z axes is the inverse (or transpose) of the rotation so that the angle sequence reverses:

$$\boldsymbol{\Omega}_{z}(\alpha)\boldsymbol{\Omega}_{x}(\beta)\boldsymbol{\Omega}_{z}(\gamma) \tag{1.8}$$

This is *Euler angle z–y–z orientation convention*.

3.2.3.2 Euler angles in the x-y-z convention (Tait-Bryan angles)

In this convention the line of nodes is the intersection of the xy-plane and $\tilde{y}\tilde{z}$-plane. The *Euler angles* in this convention are defined as follows:

ϕ is the angle between the line of nodes and the $\tilde{y}$-axis,
θ is the angle between z-axis and the $\tilde{y}\tilde{z}$-plane, and
ψ is the angle between the y-axis and the line of nodes.

These three angles specify a rotation as principal rotations about the space-fixed principal axes. The first rotation is by angle ϕ about the x-axis. The second is by angle θ about the y-axis. The third is by angle ψ about the z-axis. The combined rotation

$$\boldsymbol{R}_{z}(\psi)\boldsymbol{R}_{y}(\theta)\boldsymbol{R}_{x}(\phi) \quad \text{space-fixed.} \tag{1.9}$$

is the *Euler angle z–y–x rotation convention*. The equivalent body-fixed specification is:

$$\boldsymbol{R}_{\tilde{x}}(\phi)\boldsymbol{R}_{\tilde{y}}(\theta)\boldsymbol{R}_{\tilde{z}}(\psi) \quad \text{body-fixed.} \tag{1.10}$$

The corresponding *Euler angle x–y–z orientation convention* is the inverse operation:

$$\begin{aligned} &\boldsymbol{\Omega}_x(\phi)\boldsymbol{\Omega}_y(\theta)\boldsymbol{\Omega}_z(\psi) \quad \text{space-fixed} \\ &\boldsymbol{\Omega}_{\tilde{z}}(\psi)\boldsymbol{\Omega}_{\tilde{y}}(\theta)\boldsymbol{\Omega}_{\tilde{x}}(\phi) \quad \text{body-fixed} \end{aligned} \tag{1.11}$$

The Euler angles in this convention are variously called *Tait-Bryan angles*, *Cardano angles*, or *nautical angles*. The various names given to these angle symbols include:

ϕ roll or bank or tilt,
θ pitch or elevation, and
ψ yaw or heading or azimuth.

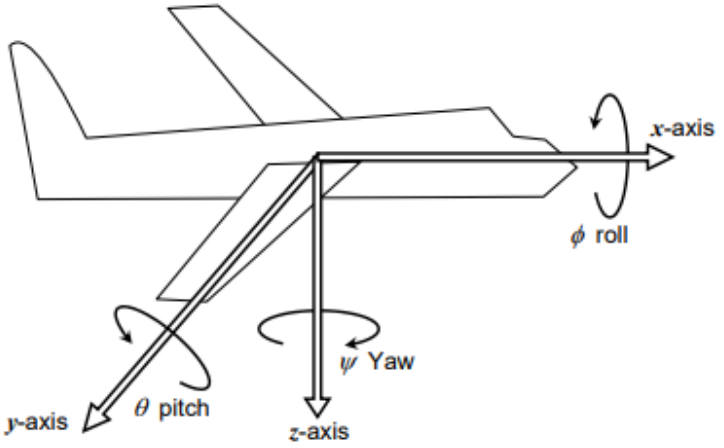


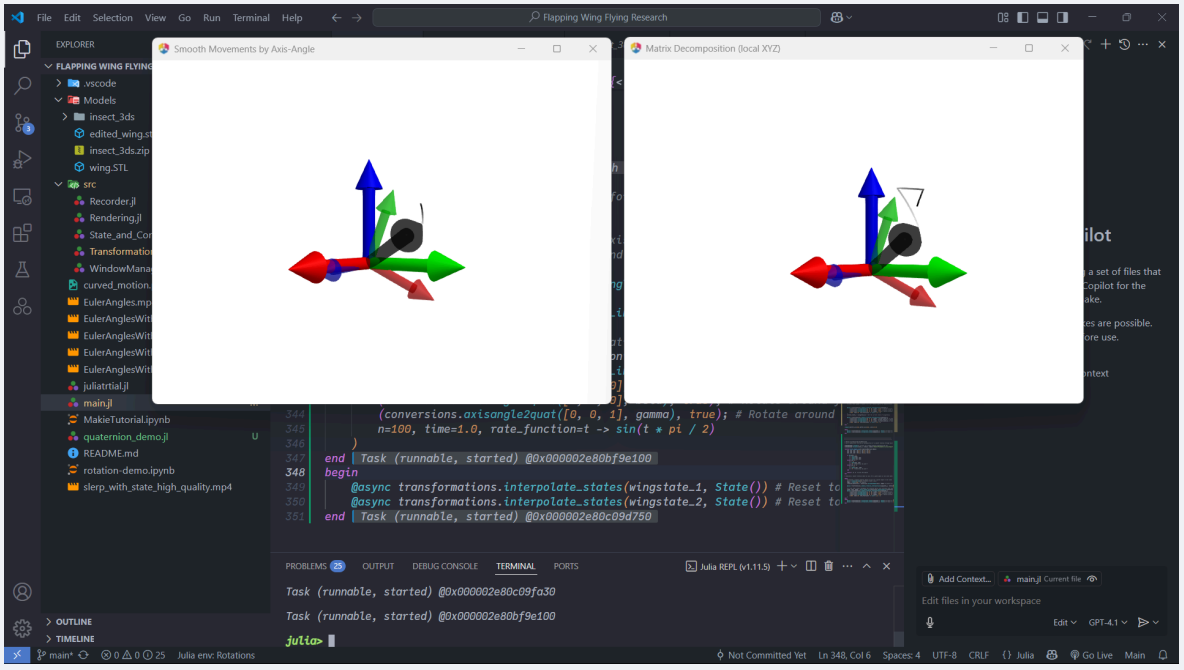
Figure 7 - Tait-Bryan angles

from what I remember, we use a left handed system where the roll and pitch are the same but the yaw is about a z-axis facing "upwards".

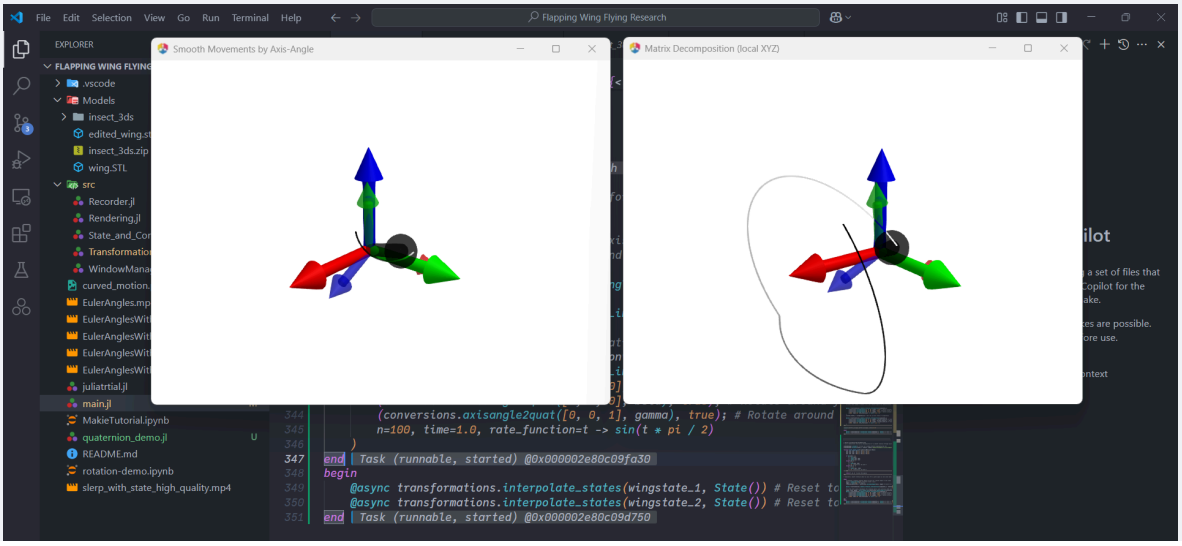
🌐https://en.wikipedia.org/wiki/Euler_angles
This is another good resource on the topic and it very nicely gives us a table

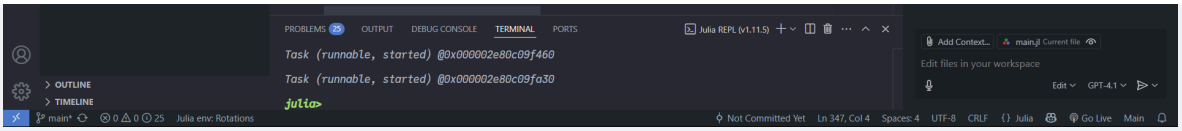
Proper Euler angles		Tait–Bryan angles	
$X_\alpha Z_\beta X_\gamma$	$\alpha = \arctan\left(\frac{R_{31}}{R_{21}}\right)$ $\beta = \arccos(R_{11})$ $\gamma = \arctan\left(\frac{R_{13}}{-R_{12}}\right)$	$X_\alpha Z_\beta Y_\gamma$	$\alpha = \arctan\left(\frac{R_{32}}{R_{22}}\right)$ $\beta = \arcsin(-R_{12})$ $\gamma = \arctan\left(\frac{R_{13}}{R_{11}}\right)$
$X_\alpha Y_\beta X_\gamma$	$\alpha = \arctan\left(\frac{R_{21}}{-R_{31}}\right)$ $\beta = \arccos(R_{11})$ $\gamma = \arctan\left(\frac{R_{12}}{R_{13}}\right)$	$X_\alpha Y_\beta Z_\gamma$	$\alpha = \arctan\left(\frac{-R_{23}}{R_{33}}\right)$ $\beta = \arcsin(R_{13})$ $\gamma = \arctan\left(\frac{-R_{12}}{R_{11}}\right)$
$Y_\alpha X_\beta Y_\gamma$	$\alpha = \arctan\left(\frac{R_{12}}{R_{32}}\right)$ $\beta = \arccos(R_{22})$ $\gamma = \arctan\left(\frac{R_{21}}{-R_{23}}\right)$	$Y_\alpha X_\beta Z_\gamma$	$\alpha = \arctan\left(\frac{R_{13}}{R_{33}}\right)$ $\beta = \arcsin(-R_{23})$ $\gamma = \arctan\left(\frac{R_{21}}{R_{22}}\right)$
	$\alpha = \arctan\left(\frac{R_{32}}{-R_{12}}\right)$		$\alpha = \arctan\left(\frac{-R_{31}}{R_{11}}\right)$

$Y_\alpha Z_\beta Y_\gamma$	$\beta = \arccos(R_{22})$ $\gamma = \arctan\left(\frac{R_{23}}{R_{21}}\right)$	$Y_\alpha Z_\beta X_\gamma$	$\beta = \arcsin(R_{21})$ $\gamma = \arctan\left(\frac{-R_{23}}{R_{22}}\right)$
$Z_\alpha Y_\beta Z_\gamma$	$\alpha = \arctan\left(\frac{R_{23}}{R_{13}}\right)$ $\beta = \arctan\left(\frac{\sqrt{1-R_{33}^2}}{R_{33}}\right)$ $\gamma = \arctan\left(\frac{R_{32}}{-R_{31}}\right)$	$Z_\alpha Y_\beta X_\gamma$	$\alpha = \arctan\left(\frac{R_{21}}{R_{11}}\right)$ $\beta = \arcsin(-R_{31})$ $\gamma = \arctan\left(\frac{R_{32}}{R_{33}}\right)$
$Z_\alpha X_\beta Z_\gamma$	$\alpha = \arctan\left(\frac{R_{13}}{-R_{23}}\right)$ $\beta = \arccos(R_{33})$ $\gamma = \arctan\left(\frac{R_{31}}{R_{32}}\right)$	$Z_\alpha X_\beta Y_\gamma$	$\alpha = \arctan\left(\frac{-R_{12}}{R_{22}}\right)$ $\beta = \arcsin(R_{32})$ $\gamma = \arctan\left(\frac{-R_{31}}{R_{33}}\right)$



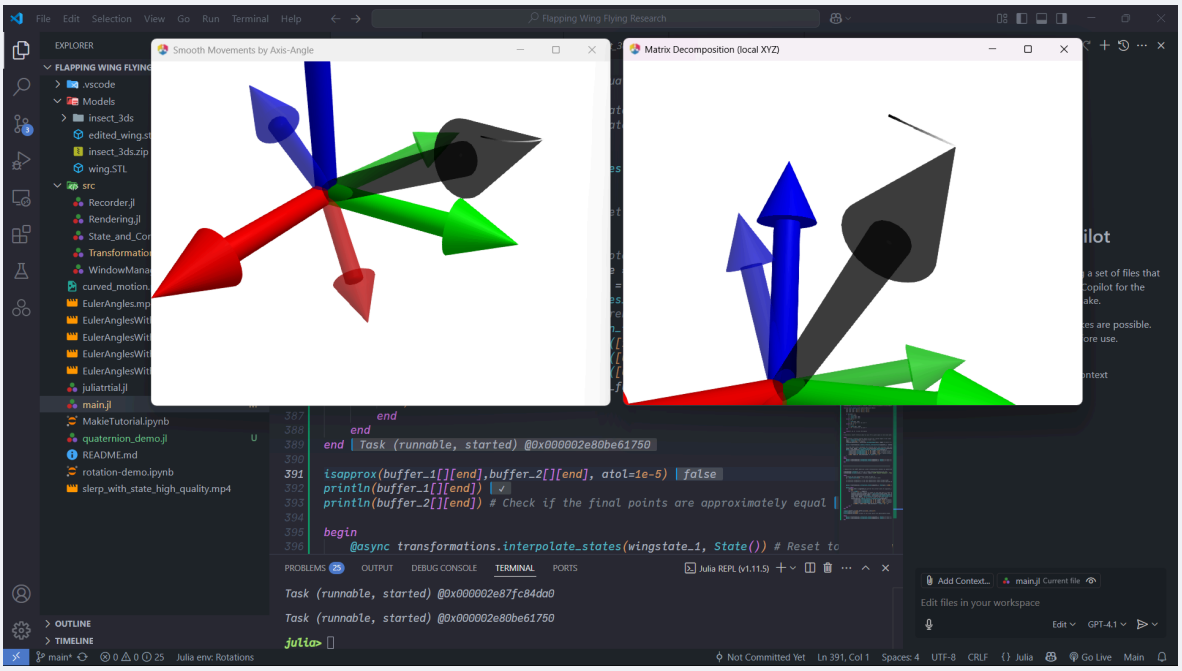
works really well.



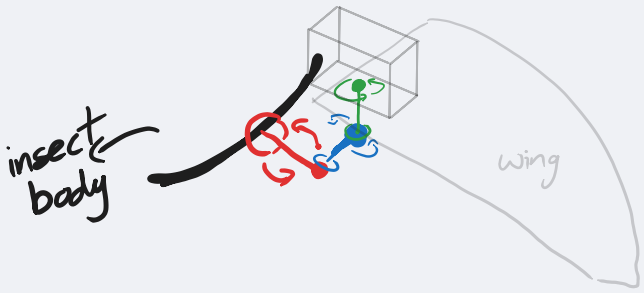


such cases are difficult to handle.

Fortunately, for small rotations, I hope that since rotations are SO(3) groups, there will be no big issues.



More granularity does improve the jaggedness of the path. But lets see if we can do any better:



From what I can tell, getting to know more about the movement constraints can help us model more effectively, being able to control different angles at the same time

A ligament like movement mechanism can also open up pathways to move along axes other that those of the gimbal.