502 Controller

Equations of motion for the interesting parts:

$$
\begin{aligned}
\ddot{x} &= \frac{(\sin(\phi)\ \sin(\psi) + \cos(\phi)\ \cos(\psi)\ \sin(\theta))\ (u_1 + u_2 + u_3 + u_4)}{m} \\
\ddot{y} &= -\frac{(\cos(\psi)\ \sin(\phi) - \cos(\phi)\ \sin(\psi)\ \sin(\theta))\ (u_1 + u_2 + u_3 + u_4)}{m} \\
\ddot{z} &= \frac{\cos(\phi)\ \cos(\theta)\ (u_1 + u_2 + u_3 + u_4)}{m} - g \\
\dot{\omega}_1 &= \frac{l\,u_2 - l\,u_4 + I_{22}\,\omega_2\,\omega_3 - I_{33}\,\omega_2\,\omega_3}{I_{11}} \\
\dot{\omega}_2 &= -\frac{l\,u_1 - l\,u_3 + I_{11}\,\omega_1\,\omega_3 - I_{33}\,\omega_1\,\omega_3}{I_{22}} \\
\dot{\omega}_3 &= \frac{\sigma\,u_1 - \sigma\,u_2 + \sigma\,u_3 - \sigma\,u_4 + I_{11}\,\omega_1\,\omega_2 - I_{22}\,\omega_1\,\omega_2}{I_{33}}
\end{aligned}
$$

Common approach seems to be grouping the control inputs together in the following order:

$$
\begin{aligned}
U_1 &= \frac{u_1 + u_2 + u_3 + u_4}{m} \\
U_2 &= \frac{l(u_2 - u_4)}{I_{11}} \\
U_3 &= -\frac{l(u_1 - u_3)}{I_{22}} \\
U_4 &= \frac{\sigma(u_1 - u_2 + u_3 - u_4)}{I_{33}}
\end{aligned}
$$

For these, we can rewrite these as a matrix multiplication

$$
\begin{bmatrix}
\frac{1}{m} & \frac{1}{m} & \frac{1}{m} & \frac{1}{m} \\
0 & \frac{l}{I_{11}} & 0 & -\frac{l}{I_{11}} \\
-\frac{l}{I_{22}} & 0 & \frac{l}{I_{22}} & 0 \\
\frac{\sigma}{I_{33}} & -\frac{\sigma}{I_{33}} & \frac{\sigma}{I_{33}} & -\frac{\sigma}{I_{33}}
\end{bmatrix}
\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}
=
\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix}
$$

This is full rank, so it is invertible, which is of the following form:

$$
\begin{pmatrix}
\frac{1}{4\,m} & 0 & \frac{1}{2\,l_{22}} & \frac{1}{4\,l_{33}} \\
\frac{1}{4\,m} & \frac{1}{2\,l_{11}} & 0 & -\frac{1}{4\,l_{33}} \\
\frac{1}{4\,m} & 0 & -\frac{1}{2\,l_{22}} & \frac{1}{4\,l_{33}} \\
\frac{1}{4\,m} & -\frac{1}{2\,l_{11}} & 0 & -\frac{1}{4\,l_{33}}
\end{pmatrix}
$$

Given this inverse, we can calculate the individual values of the control inputs g\frac{\partial }{\partial t}iven the signals.

Which I verified and obtained through the following matlab script:

```
syms m l11 l22 l33 'positive'


mat = [ m   m   m m;
         0 l11   0 -l11;
       l22   0  -l22 0;
       l33 -l33 l33 -l33;];

% Prints out 4
rank(mat)
% Prints out a valid
latex(inv(mat))
```

This (along with some rearranging) transforms the above equations:

$$\ddot{z} = U_1 \frac{\cos(\phi)\ \cos(\theta)}{m} - g$$
$$\dot{\omega}_1 = U_2 + \omega_2\,\omega_3\,\frac{I_{22} - I_{33}}{I_{11}}$$
$$\dot{\omega}_2 = U_3 - \omega_1\,\omega_3\,\frac{I_{11} - I_{33}}{I_{22}}$$
$$\dot{\omega}_3 = U_4 + \omega_1\,\omega_2\,\frac{I_{11} - I_{22}}{I_{33}}$$

Now, to begin devising the controller, the overall structure of the controller is as follows:

$$U_i(t) = U_{eq}(t) + U_D(t)$$

U_D is the drawing phase of the SMC, and U_{eq} is the sliding phase.

We can write the drawing phase part as:

$$U_D = k_D\,\mathrm{sign}(s(t))$$

However, this has a lot of problems with chattering near the sliding surface, so we can rewrite this as:

$$U_D = k_D\,\frac{s(t)}{|s(t)| + \delta}$$

This makes it behave like a proportional controller near the sliding surface, and saturated at k_D elsewhere. Both k_D and δ are tuned parameters, which we can start at 1 and then tune.

$s(t)$ is the sliding surface, which we can start out with as

$$s(t) = \dot{e} + \lambda e$$

Where e is the error of the desired characteristic. Because we have only four actuators, we can only control 4 degrees of freedom. For this, we can control z, ϕ, θ, ψ.

We can start with deriving the control law for z:

$$\begin{aligned} e &= z_d - z \\ s &= (\dot{z}_d - \dot{z}) + \lambda(z_d - z) \end{aligned}$$

The system is in a sliding condition when $\dot{s} = 0$, we can find $\dot{s}$:

$$\dot{s} = (\ddot{z}_d - \ddot{z}) + \lambda(\dot{z}_d - \dot{z})$$

We can substitute in our equation for $\ddot{z}$:

$$\dot{s} = (\ddot{z}_d + g - U_1 \frac{\cos(\phi)\ \cos(\theta)}{m} - g) + \lambda(\dot{z}_d - \dot{z})$$

When the system is in sliding condition ($\dot{s} = 0$) $U_d = 0$, and therefore $U_{eq} = 0$, so we can substitute it in:

$$\begin{aligned} 0 &= (\ddot{z}_d + g - U_{eq} \frac{\cos(\phi)\ \cos(\theta)}{m}) + \lambda(\dot{z}_d - \dot{z}) \\ U_{eq} &= (\ddot{z} + g + \lambda(\dot{z}_d - \dot{z})) \frac{m}{\cos(\phi)\ \cos(\theta)} \end{aligned}$$

So we can write the full control law for z as follows:

$$U_1(t) = (\ddot{z} + g + \lambda(\dot{z}_d - \dot{z})) \frac{m}{\cos(\phi)\ \cos(\theta)} + k_D \frac{s(t)}{|s(t)| + \delta}$$

The other terms can be similarly solved for. Note that because I am controlling ω, the euler angles are in reference to the body fixed frame, which will be denoted with a $_b$ notation.

WE CAN just convert the 0 angle into the body frame at each timestep so that we can just deal with the body frame accelrations and other such stuff, no need to complicate it a lot.
We can also do the same process for the desired angular acceleration/velocity.

Solving for the other Control laws:

$$\begin{aligned} e &= \phi_{b_d} - \phi_b \\ s &= (\dot{\phi}_{b_d} - \omega_1) + \lambda(\phi_{b_d} - \phi_b) \\ \dot{s} &= (\ddot{\phi}_{b_d} - \dot{\omega}_1) + \lambda(\dot{\phi}_{b_d} - \omega_1) \\ &= (\ddot{\phi}_{b_d} - U_2 - \omega_2\,\omega_3\,\frac{I_{22} - I_{33}}{I_{11}}) + \lambda(\dot{\phi}_{b_d} - \omega_1) \\ \implies U_{eq\phi} &= (\ddot{\phi}_{b_d} - \omega_2\,\omega_3\,\frac{I_{22} - I_{33}}{I_{11}}) + \lambda(\dot{\phi}_{b_d} - \omega_1) \end{aligned}$$

$$
\begin{aligned}
e &= \theta_{b_d} - \theta_b \\
s &= (\dot{\theta}_{b_d} - \omega_2) + \lambda(\theta_{b_d} - \theta_b) \\
\dot{s} &= (\ddot{\theta}_{b_d} - \dot{\omega}_2) + \lambda(\dot{\theta}_{b_d} - \omega_2) \\
&= (\ddot{\theta}_{b_d} - U_3 + \omega_1\,\omega_3\,\frac{I_{11} - I_{33}}{I_{22}}) + \lambda(\dot{\theta}_{b_d} - \omega_2) \\
\implies U_{eq\theta} &= (\ddot{\theta}_{b_d} + \omega_1\,\omega_3\,\frac{I_{11} - I_{33}}{I_{22}}) + \lambda(\dot{\theta}_{b_d} - \omega_2)
\end{aligned}
$$

$$
\begin{aligned}
e &= \psi_{b_d} - \psi_b \\
s &= (\dot{\psi}_{b_d} - \omega_3) + \lambda(\psi_{b_d} - \psi_b) \\
\dot{s} &= (\ddot{\psi}_{b_d} - \dot{\omega}_3) + \lambda(\dot{\psi}_{b_d} - \omega_3) \\
&= (\ddot{\psi}_{b_d} - U_4 - \omega_1\,\omega_2\,\frac{I_{11} - I_{22}}{I_{33}}) + \lambda(\dot{\psi}_{b_d} - \omega_3) \\
\implies U_{eq\psi} &= (\ddot{\psi}_{b_d} - \omega_1\,\omega_2\,\frac{I_{11} - I_{22}}{I_{33}}) + \lambda(\dot{\psi}_{b_d} - \omega_3)
\end{aligned}
$$

So for these, we need to find the desired position, velocity, and acceleration. There are two ways to do this: At each step, we generate a trajectory to our target point, which gives us those quantities, or we use some kind of PID controller to generate a signal that gets us to the point, which should work well enough for the kinematics. It's probably easier to just do the trajectory.

The other problem is that we can only really generate a trajectory in linear space, so how to we translate that into movements in the angles?

Also for the angles, we will have to set desired states in the inertial frame, and then translate that into the body frame. That shouldn't be too bad. I'm not sure how to do position, velocity is outlined in the quadrotor dynamics pdf, and for acceleration you can just take the derivative of that stuff:

$$
T = \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \sin(\phi)\cos(\theta) \\ 0 & -\sin(\phi) & \cos(\phi)\cos(\theta) \end{bmatrix}
$$

$$
\dot{a} = T\omega
$$

$$
\ddot{\alpha} = \dot{T}\omega + T\dot{\omega}
$$

For position, chatgpt gave me this:
To map an orientation defined in Euler angles in one coordinate frame to the same orientation in another coordinate frame, you typically follow these steps:

1. **Convert Euler Angles to a Rotation Matrix**: First, convert the Euler angles defining the orientation in the original frame to a rotation matrix. Euler angles can be converted to a rotation matrix using specific sequences, such as ZYX, XYZ, etc., depending on the convention used (intrinsic or extrinsic rotations).

2. **Apply a Transformation Matrix**: If the second frame is related to the first frame through a known transformation (rotation and/or translation), apply this transformation to the rotation matrix. This transformation matrix aligns the original frame with the new frame.
3. **Convert Back to Euler Angles**: Finally, convert the transformed rotation matrix back into Euler angles using the appropriate sequence and convention that applies to the second frame.

This process allows you to maintain the same orientation relative to the respective coordinate frames, even if the frames themselves are oriented differently in space.

So using all of that, we can get appropriate desired states for the problem.