ME 181 Advanced Dynamics

Final Project

Cen, Jiajin   Chen, Longye   Edegware, Michael   Etkind, Josh

Introduction

The relatively recent explosion of popularity in quadrotor drones has created a thriving community of manufacturers, hobbyists, and commercial users. New extremely lightweight Lithium Ion batteries coupled with lighter, more powerful electric motors has given these platforms unique aerobatic capabilities. As their popularity has grown, an exciting new sport has developed: drone racing. This paper investigates how well-understood principles of dynamics apply to drone racing. We use Simscape Multibody to create a physical model of a drone and simulate its motion through a course consisting of four checkpoints in the shape of hoops. Control of the drone is achieved by torques commands transmitted to the motor models. We begin by completing the course and then investigate ways to improve the speed at which we complete the course, including the effect of the moment of inertia of the central body of the drone as well.

After our effort, our drone can cross all the four hoops in 8.22s!

Background

The key idea of this project was to use Simscape Multibody to simulate a quadrotor drone racing to complete an obstacle course in the shortest time possible by flying through four hoops in various positions and orientations. We modeled a quadrotor with nine parts (Figure 1): a central body, four motor housings and four rotors. To steer the drone through the course we constructed control commands (motor thrust vs. Time) and routed them to our Simulink motor models. Lu, J., & Smith, B. (2017) provide an overview of relevant mathematics and assumptions of drone racing from a general and accessible perspective.

Completion of the course is defined as the instant at which the drone successfully navigates its center through the plane of the final ring. Since the drone must visit all rings once and only once, the problem of navigating to each rin can be conceived as a simple version of the Travelling Salesman problem. Kim, S., & Moon, I. (2019, January) describe an application of the travelling salesman problem to routing a drone through a small number of waypoints.

A few rules are imposed on the race we will simulate. First, only the mass distribution within the central sphere may be changed. All other masses and distributions must remain the same. Additionally, should the distribution of mass within the central body be changed, it must remain between $0.1mr^2$and $0.9mr^2$.Finally, the maximum torque that can be applied to any motor at any time is 0.12 N*m.

Methodology

Design

The quadrator's body was modeled as a rigidly connected set of five spheres. A large central sphere represented the cowling around the battery and electronics and the four smaller spheres represented the motor housings. All the four spheres are arranged to lay in the body *x-y* plane as shown in Figure 1. The smaller spheres were patterned in a square and the larger sphere was placed at the center of the square. All spheres involved in this modeling were assumed to be solid with uniform density. The numerical properties for components involved in this design were:

- Diagonal of square pattern: 30 cm
- Radius of large sphere: 5 cm
- Radius of small spheres: 2 cm
- Mass of large sphere: 1.6 kg
- Mass of each small sphere: 0.1 kg

The above assumptions and numerical properties were used to compute the body's inertia matrix shown Appendix A.

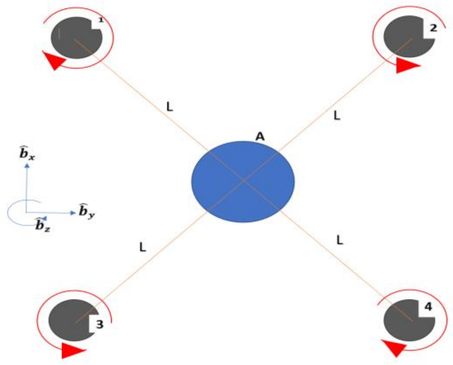


Figure 1: Illustration of drone setup with coordinate system, motor naming scheme, and propellor rotation directions

A propeller was attached to each motor such that it could freely rotate about the body z-axis. Each was represented by small disc with the following properties:

- Radius of propeller: 7.5 cm
- Thickness of propeller: 1 mm
- Mass of propeller: 0.02 kg

It was assumed that the inertial properties of the propellers have been slightly adjusted to account for the inertia of the propeller shaft and the rotating components of the motor. The inertia of the propeller was calculated as in Appendix A.

Method and rationale for Defining Motor Torques and Torque Mapping

Each motor's torque signal begins with a baseline torque meant to generate sufficient lift to balance gravity so that the drone's motion is under our control. This baseline torque is a function of the angle θbetween the $\hat{b}_z$vector in the Body frame and the $\widehat{w_z}$vector in the World frame. The baseline torque is set to the expression $\frac{0.1}{\sqrt{cos(\theta)}}$to balance gravity (see appendix B for derivation). Since motors 1 and 4 rotate in the $-\widehat{b_z}$direction, the baseline torque is negated before being routed to these motors.

3 additional torques signals were generated to move the drone in the $\widehat{b_x}$, $\widehat{b_y}$, and $\widehat{b_z}$directions (denotes u, v, and w respectively). These signals were designed to alter the angle of the drone's body in the world frame so that the drone would move in the desired direction. For example: to move in the $\widehat{b_x}$ direction a signal was first generated that initiated a net torque on the drone about its $\widehat{b_y}$axis. Then this signal was removed to cease angular acceleration about this axis. Next a signal was generated to initiate a net torque in the opposite direction so that the body would experience angular acceleration in the opposite direction, making the body to cease rotating about this axis. To move in the $\widehat{b_y}$ direction a similar signal was generated rotating the body about its $\widehat{b_x}$axis. To move in the $\widehat{b_z}$direction, the signals were designed to

either uniformly increase or decrease the torques of all motors so that the drone would accelerate in the $\widehat{b_z}$direction.

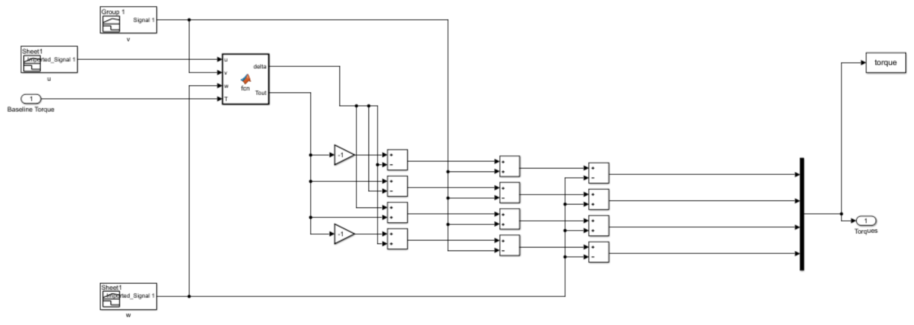


Figure 2: Signal processor

Mapping these these *u*, *v*, and *w* signals appropriately onto each motor was necessary to actually achieve the desired rotation. To rotate around the $\widehat{b_x}$axis, the *v* signal should increase the torques on motors 1 and 3 while decreasing the torques on motors 2 and 4. Motor 1 spins in the negative $\widehat{b_z}$ direction, so *v* was negated before being added to the baseline torque in order to increase the torque on this motor. Motor 3 rotates in a positive $\widehat{b_z}$sense, so to increase its torque *v* was simply added to its composite torque signal. Similarly, motor 2 spins in the positive $\widehat{b_z}$ direction, so *v* was subtracted from its signal, while motor 4 has a negative $\widehat{b_z}$ rotation rate so +*v* was added to its signal to reduce its torque. A similar process was repeated for rotation about the $\widehat{b_y}$ axis. The torque mapping is summarized in Table 1 below

Table 1: Torque mapping scheme

Motor #	Rotation Sense	For $+\widehat{b_x}$rotation, add:	For $+\widehat{b_y}$rotation, add:
1	$-\widehat{b_z}$	-*v*	+*u*
2	$+\widehat{b_z}$	+*v*	-*u*
3	$+\widehat{b_z}$	-*v*	+*u*
4	$-\widehat{b_z}$	+*v*	-*u*

One issue with this approach of simply summing various signals is that there is the potential for simultaneous signals to interfere and change the dynamics of the drone's body in unexpected ways. For example, the baseline torque signal which changes with θwill interfere with signals sent during $\widehat{b_x}$ and $\widehat{b_y}$rotation change maneuvers. To deal with this particular issue, we implemented a MATLAB function to ensure that if a signal is attempting to maneuver the drone (i.e. any of *u*, *v*, or *w* are non-zero) the baseline torque is set to 0.1 N-m – the torque required to balance gravity without any tilt. This way, the signal maneuvering the drone will be the only factor changing during the duration of the signal's impulse. If the baseline torque were to change during a signal's impulse, the changes in angular acceleration about the body generated by the alternating peaks of the signal wouldn't balance each other and the drone would have non-zero angular velocity and would enter a spin about its $\widehat{b_x}$ or $\widehat{b_y}$ axis.

Method and Rationale for Control

Our approach to routing the drone through the course was entirely iterative: we built the u, v, and w signals in spreadsheets and imported them into Simulink. To modify the trajectory of the drone, we changed the timing, duration, and internal delay in the signals. By repeatedly running the simulation and modifying the signals, we built a trajectory that passed through each hoop. No feedback or other control systems were used.

Method for Checking Trajectory

Each hoop is defined in the provided environment file as a ring with an outer radius of 0.5 m. The center of each ring is defined by Rigid Transforms along the world's cartesian coordinates from the origin of the World coordinate frame. By summing the translation of each cartesian translation leading to a given ring, its position vector relative to the world's origin can be obtained, denoted here as the vector $\underline{r^{i/w_0}}$for each ring $i = \{1, 2, 3, 4\}$. Each ring also has a directional sense defined by the vector normal to the plane in which its mass is primarily distributed or equivalently a vector perpendicular to two distinct vectors from the ring's center to its edge. This direction will be defined as the unit vector normal to this plane $\widehat{n_i}$. In this particular environment, all rings happen to be oriented with $\widehat{n_i}$aligned with one of the World frame's basis vectors.

During the simulation, the position of the center of the drone's body relative to the origin of the World coordinate frame is measured in the world x, y, and z directions using a Transform Sensor block at each time step of the simulation, denoted herein as the set of vectors $r^{B_{cm}/w_0}(t)$. The position of the drone at each timestep relative to each hoop's center can then be calculated as the set of vectors $r^{B_{cm}/i}(t) = r^{i/w_0} - r^{B_{cm}/w_0}(t)$.

The conditions for a successful pass through a given hoop are defined as the center of the drone's body sphere passing through the confines of the hoop by beginning at one side of the hoop at one time step and ending on the other side at the next time step. This definition suggests two criteria that define a successful pass. First, the drone must begin on one side of the hoop and end on the other. This condition is modeled here as the dot product of a given hoop's normal vector and the drone's position relative to that hoop ($\widehat{n_i} \cdot r^{B_{cm}/i}$) changing signs. This can be thought of as the measure of the drone's position along the axis of the hoop changing from being in the same direction as the normal vector to being in the opposite. If this condition is met, the drone has crossed the plane of the ring. Second, when the drone crosses the plane of the ring it must have done so close enough to the ring's center to be within the enclosure of the ring. The distance between the ring's center and the center of mass of the drone is modeled as the magnitude of the drone's position relative to a given hoop being no greater than one hoop radius, $||r^{B_{cm}/i}|| <= r_{hoop}$, or $||r^{B_{cm}/i}|| <= 0.15$m . If both these conditions are met then the drone has crossed the plane of the hoop and done so within the bounds of the hoop's radius, therefore passing through the hoop.

Method for Tuning/Optimizing Trajectory Design to Reduce Time-to-Finish

Our first pass at completing the course consisted of navigating entirely in cartesian directions: first moving in the $\widehat{b_x}$ direction, stopping, then moving in the $\widehat{b_z}$direction, and again stopping. We continued to navigate the course in this rudimentary way until we could successfully fly through each hoop. Next, our task was to increase the efficiency of our route and attempt to minimize the time required to navigate the course.

We generated acceleration against the direction of the drone's motion by adding different pairs of pulses to leading and trailing propellers so that the unwanted movement in XY direction could be minimized. In Z direction, we reduced the duration of the pulses as much as possible to ensure the drone didn't fly too high.

By moving in more than 1 dimension at a time, we were able to significantly reduce the minimum time required to navigate the course, winding up at 8.22 seconds. Our navigation strategy was as follows:

1. Move in positive x-direction horizontally and pass through the first hoop whose center is at (5,0,0)
2. Continue forward in the +x direction while moving upward in the positive z-direction to cross the second hoop which is at (10,0,2)
3. Fly towards the third hoop with the coordinates (35,10,5) in all directions
4. Dive in a curve to the forth hoop at (35,0,0) and cross it to finish the course.

Results

Trajectory of Fastest Run

Our drone model was able to successfully navigate the course in 8.22 s. The trajectory it followed is plotted in 3 dimensions below in Figure 4. Figure 5 illustrates the drone finishing the course, including the simulation timestamp in the bottom right hand corner.

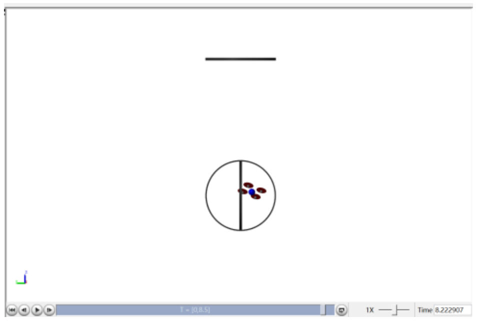


Figure 3. View of simulation after drone completes course. Note simulation time of 8.222907 s.

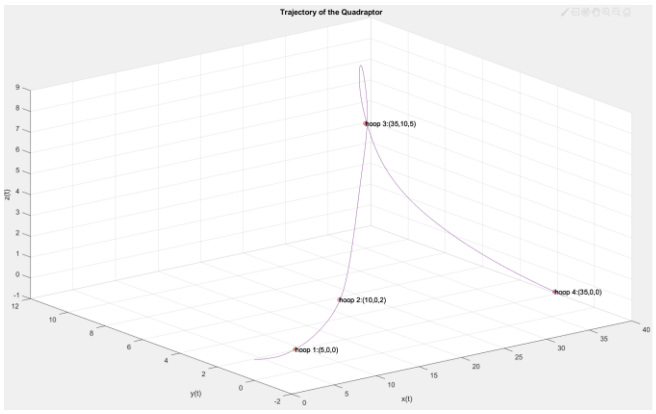


Figure 4: 3D plot of trajectory including markers and coordinates of each hoop

Verifying Fastest Run

As discussed in the Methodology section, the drone is considered to pass through a hoop when the dot product of its position vector relative to the hoop and the hoop's normal vector changes signs while the drone is within 1 hoop radius or 1 m from the center of the hoop. From Figure 7, we can see that the drone passed through hoop 1 near *t*=1.7 s, hoop 2 near *t*=2.2 s, hoop 3 near *t*=6.1 s, and finished the course by passing through hoop 4 near *t*=8.2 s. These results agree with the time that the drone appeared to cross the final hoop as seen in Figure 5.

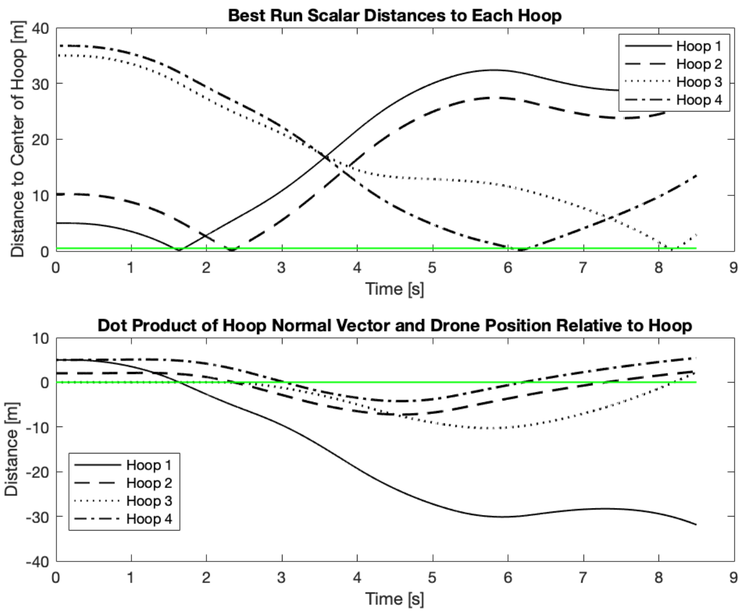


Figure 5: Verification of passing through four hoops

After verifying that the drone did complete the course, it was then necessary to be sure that the drone didn't violate the rules of the race. The maximum torque that any motor of the drone was allowed to experience at any given time was 0.12 N*m. Figure 8 verifies that at no time did the drone violate this condition.

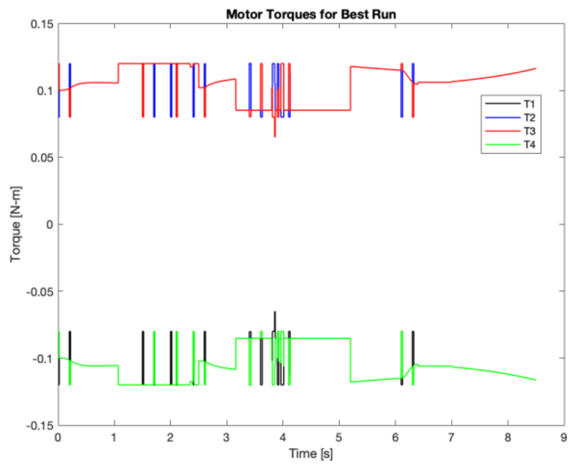


Figure 6: Verification of no exceeding limit torque

System Mass Verification

To verify that the mass of our drone was accurately represented in our simulation, we performed a test in which we simply programmed each motor's torque to the maximum allowable value, 0.12 N*m. Figure 8 illustrates its acceleration curve as well as an analytical model of an acceleration curve explained in further detail in Appendix D. Note that the theoretical model does not include drag, which is why it comes to an asymptote are 4.3 m/s^2. Our drone model includes drag, which explains why its acceleration slopes downwards after time instead of following the same asymptote as the model.

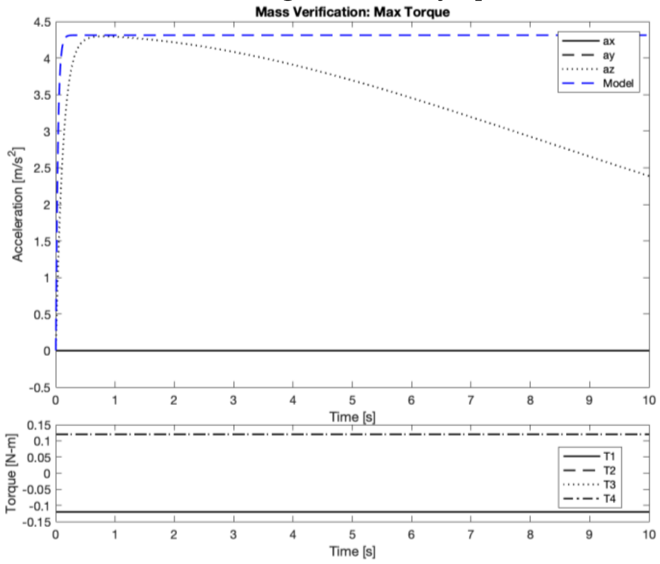


Figure 7: Acceleration from rest with maximum torque on all motors.

Optional part: Power lost

After collecting data from the simulation, we calculated the power lost with the equation $P = Fv$ and $P = Mw$ by the code shown in Appendix C. The result is shown below.

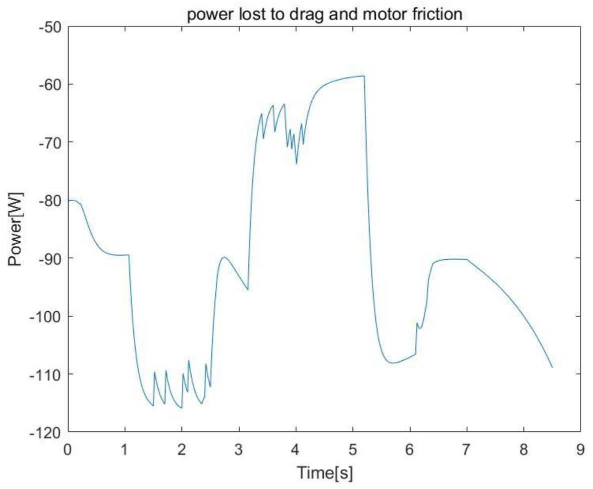


Figure 8: Power lost to drag and motor friction

Conclusions

After our quadrotor finishes its flight, we can conclude the approaches and key results. Our first approach is to simplify the simulink model so that we can give signal to four motors at the same time instead of one signal for one motor. We also give the quadrotor a torque to make it have a constant velocity in z-direction when no other torque is applied on it to simplify our model more. Then we separated the signal input into three parts to control the acceleration in x, y and z direction separately. After testing and adjusting our model step by step, we finally got a successful model to cross the four hoops.With that bodel, we used basic vector geometry to verify that the drone passed through each hoop. For the ‘power lost’ part, we used method P=Fv and P=Mw to calculate the power lost to drag and motor friction separately.

In conclusion, our key results are:

1. Time to finish: 8.22s;
2. Successfully cross all four hoops;
3. The torque never exceeded the limit;
4. The power lost is shown in Figure 8.

References

Lu, J., & Smith, B. (2017). Autonomous navigation in drone racecourse. Retrieved May 2, 2019, from https://ieeexplore.ieee.org/document/8284213

Kim, S., & Moon, I. (2019, January). Traveling Salesman Problem With a Drone Station. Retrieved May 2, 2019, from https://ieeexplore.ieee.org/document/8488559

Appendices

Appendix A: Analysis of Inertia

If:

- L = Half the diagonal of square pattern
- $\boldsymbol{R}$ = Radius of large sphere
- $\boldsymbol{r}$ = Radius of small spheres
- $\boldsymbol{M}$ = Mass of large sphere
- $\boldsymbol{M}$ = Mass of each small sphere
- r_p = Radius of propeller
- **h** = Thickness of propeller
- m_p =Mass of propeller

Then,

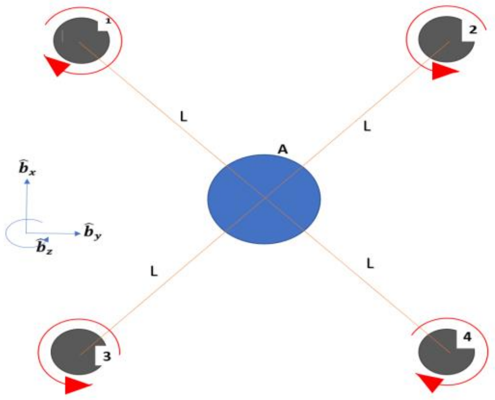


Figure A-1. Schematics of the quadrator

From figure A-1, the inertia about A's own center is:

$$\overline{\overline{I^{A/A_{cm}}}} = \frac{2}{5}\boldsymbol{MR^2}\begin{pmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{pmatrix}_{\boldsymbol{b}}$$

Similarly, Let the motor 2 be D. Its inertia about its center is:

$$\overline{\overline{I^{D/D_{cm}}}} = \frac{2}{5}\boldsymbol{mr^2}\begin{pmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{pmatrix}_{\boldsymbol{b}}$$

However, we need the inertia matrix about the the center of mass(Acm). From parallel angle theorem,

$$\overline{\overline{I^{D/A_{cm}}}} = \overline{\overline{I^{D/D_{cm}}}} + \begin{pmatrix}my^2 & -mxy & 0\\-mxy & mx^2 & 0\\0 & 0 & m(x^2+y^2)\end{pmatrix}_{\boldsymbol{b}}$$

Where:

$$x = \frac{L}{\sqrt{2}}(\hat{\boldsymbol{b}}_x + \hat{\boldsymbol{b}}_y)\cdot\hat{\boldsymbol{b}}_x = \frac{L}{\sqrt{2}} \qquad \mathrm{y} = \frac{L}{\sqrt{2}}(\hat{\boldsymbol{b}}_x + \hat{\boldsymbol{b}}_y)\cdot\hat{\boldsymbol{b}}_y = \frac{L}{\sqrt{2}}$$

Due to symmetry, the inertia of the body can be calculated as:

$$\overline{\overline{I^{B/O}}} = \overline{\overline{I^{A/A_{cm}}}} + 4\begin{pmatrix}my^2 & 0 & 0\\0 & mx^2 & 0\\0 & 0 & m(x^2+y^2)\end{pmatrix}_{\boldsymbol{b}}$$

Let P represent a propeller, therefore:

$$\overline{\overline{\boldsymbol{I}^{P/P_{cm}}}} = \frac{1}{12}\begin{pmatrix} 3(r_p)^2 + h^2 & 0 & 0 \\ 0 & 3(r_p)^2 + h^2 & 0 \\ 0 & 0 & 6(r_p)^2 \end{pmatrix}_{\boldsymbol{b}}$$

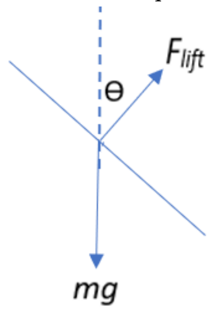
From parallel angle theorem,

$$\overline{\overline{\boldsymbol{I}^{\mathrm{P/O}}}} = \overline{\overline{\boldsymbol{I}^{\mathrm{P}/\mathrm{P}_{cm}}}} + \mathrm{m_p}\begin{pmatrix} \frac{L^2}{2} + r_p{}^2 & -\frac{L^2}{2} & \frac{\mathrm{L}\sqrt{3}}{\sqrt{2}} \\ -\frac{L^2}{2} & \frac{L^2}{2} + r_p{}^2 & \frac{\mathrm{L}\sqrt{3}}{\sqrt{2}} \\ \frac{\mathrm{L}\sqrt{3}}{\sqrt{2}} & \frac{\mathrm{L}\sqrt{3}}{\sqrt{2}} & \mathrm{L}^2 \end{pmatrix}_{\boldsymbol{b}}$$

Appendix B: Torque Necessary to Balance Gravity During Tilting

Derivation for $\tau = \frac{0.1}{\sqrt{cos(\theta)}}$

FBD for the quadrator:



At static equilibrium,
The drone will maintained a horizontal plane and moved in the Z-direction with a constant velocity,
That is,

$$F_{lift}cos(\theta) = mg$$

But $F_{lift} \propto \omega^2$ and $\tau \propto \omega$
This implies:
$F_{lift} \propto \tau^2$

$$F_{lift} = C\tau^2$$

$\frac{mg}{cos(\theta)} = C\tau^2$ substitute initial conditions: $\theta = 0$ and $\tau = 0.1$

C = 100mg
Thus,

$$\frac{mg}{cos(\theta)} = C\tau^2$$

$$\tau = \frac{0.1}{\sqrt{cos(\theta)}}$$

Appendix C: MATLAB Code for Power Loss Calculations

```
drag_0=drag0.Data;
drag_1=drag1.Data;
drag_2=drag2.Data;
drag_3=drag3.Data;
drag_4=drag4.Data;
X=vx.Data;
X1=x1.Data;
X2=x2.Data;
X3=x3.Data;
X4=x4.Data;
Y=vy.Data;
Y1=y1.Data;
Y2=y2.Data;
Y3=y3.Data;
Y4=y4.Data;
Z=vz.Data;
Z1=z1.Data;
Z2=z2.Data;
Z3=z3.Data;
Z4=z4.Data;
P_drag0=drag_0(:,1).*X+drag_0(:,2).*Y+drag_0(:,3).*Z;
P_drag1=drag_1(:,1).*X1+drag_1(:,2).*Y1+drag_1(:,3).*Z1;
P_drag2=drag_2(:,1).*X2+drag_2(:,2).*Y2+drag_2(:,3).*Z2;
P_drag3=drag_3(:,1).*X3+drag_3(:,2).*Y3+drag_3(:,3).*Z3;
P_drag4=drag_4(:,1).*X4+drag_4(:,2).*Y4+drag_4(:,3).*Z4;
P_drag=P_drag0+P_drag1+P_drag2+P_drag3+P_drag4;
W1=w1.Data;
W2=w2.Data;
W3=w3.Data;
W4=w4.Data;
K=5e-4;
P_mf=-K.*W1.*W1-K.*W2.*W2-K.*W3.*W3-K.*W4.*W4;
P_lost=P_drag+P_mf;
t=w1.Time;
figure
plot(t,P_lost)
xlabel('Time[s]')
ylabel('Power[W]')
title('power lost to drag and motor friction')
```

Appendix D: Model of Acceleration Under Maximum Torque

$$T = 0.12\ Nm$$

$$L = m_s g/4 * w^2/200^2$$
$$I = m * r^2$$
$$w(0) = 200\, rad/s$$
$$m_s = 2.08\, kg$$
$$m = 0.1\, kg$$
$$r = 0.02\, m$$
$$g = 9.80065\, m/s^2$$
$$k = 5e - 4\, Nms/rad$$

For rotation, we have:

$$T - k * w = I * w'$$

With the initial condition, we can have

$$w = -40e^{-31.25t} + 240\, rad/s$$

If we do the FBD for the drone, we can have

$$L - mg = ma_z$$

As a result, we get az:

$$a_z = (((-e^{-31.25t} + 6)/5)^2 - 1) * 9.80065$$