UCLA ENGINEERING

183 DB Weekly Report

Final Presentation

Team Parsley

Lin Li, Amir Omidfar, Wilson Chang, Angel Jimenez

parsley

Underactuated Robotics

- Interest on underactuated Robotics
- Make use of system dynamics, more natural, cost and power efficient
- Inspiration from Rocket Control, controller for steering
- Explore the possibility through a quadcopter

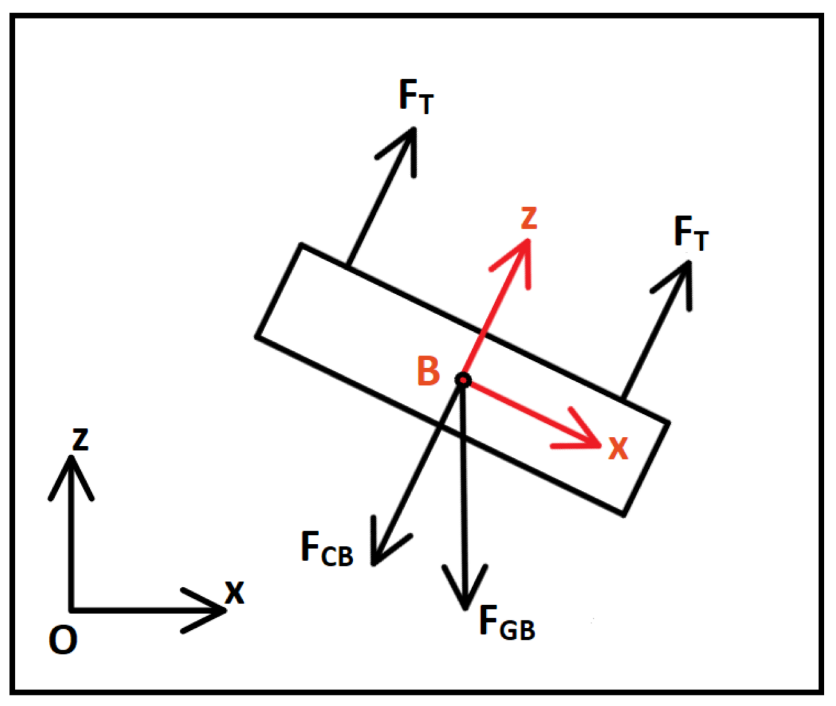
Outline

- Mathematical Model
- Simulation
- Motor Exploration
- Actual Implementation
- Conclusion and Expectation for demo

Body Model

$$^{O}\boldsymbol{F}_{\boldsymbol{net,B}} = {}^{O}\boldsymbol{F}_{\boldsymbol{GB}} + {}^{O}\boldsymbol{F}_{\boldsymbol{T}} + {}^{O}\boldsymbol{F}_{\boldsymbol{CB}} = m_B\, {}^{O}\boldsymbol{a}_{\boldsymbol{B}}$$

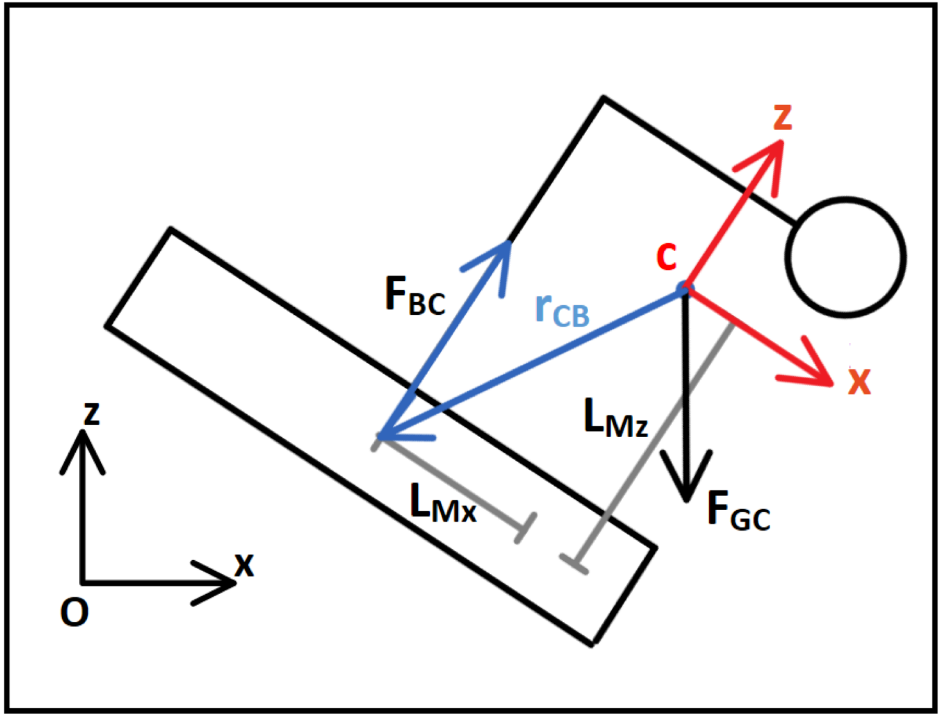
$$^{O}\boldsymbol{\tau}_{\boldsymbol{net,B}} = R(\boldsymbol{q}_{\boldsymbol{B}})\, {}^{B}\boldsymbol{\tau}_{\boldsymbol{CB}} = {}^{O}I_B\, {}^{O}\boldsymbol{\alpha}_{\boldsymbol{B}}$$



Controller Model

$$ {}^{O}\boldsymbol{F}_{\boldsymbol{net,C}} = {}^{O}\boldsymbol{F}_{\boldsymbol{BC}} + {}^{O}\boldsymbol{F}_{\boldsymbol{GC}} = m_C \, {}^{O}\boldsymbol{a}_{\boldsymbol{C}} $$

$$ {}^{O}\boldsymbol{\tau}_{\boldsymbol{net,C}} = R(\boldsymbol{q}_{\boldsymbol{C}}) \, {}^{C}\boldsymbol{\tau}_{\boldsymbol{BC}} + {}^{O}\boldsymbol{\tau}_{\boldsymbol{RF}} = {}^{O}I_c \, {}^{O}\boldsymbol{\alpha}_{\boldsymbol{C}} $$



Merging equations

- We could never know the reaction force and torque

$$ {}^{O}\boldsymbol{F}_{BC} = -{}^{O}\boldsymbol{F}_{CB} $$
$$ {}^{O}\boldsymbol{\tau}_{BC} = -{}^{O}\boldsymbol{\tau}_{CB} $$

- Angular acceleration

$$ 2\left[\ddot{q}_B q_B^* - (\dot{q}_B q_B^*)^2\right] $$
$$ 2\left[\ddot{q}_C q_C^* - (\dot{q}_C q_C^*)^2\right] $$

From Newton’s 2nd Law of Motion

$$ {}^{O}\boldsymbol{F}_{\mathbf{net},C} = {}^{O}\boldsymbol{F}_{BC} + {}^{O}\boldsymbol{F}_{GC} = m_C\,{}^{O}\boldsymbol{a}_C $$
$$ {}^{O}\boldsymbol{\tau}_{\mathbf{net},C} = R(\boldsymbol{q}_C)\,{}^{C}\boldsymbol{\tau}_{BC} + {}^{O}\boldsymbol{\tau}_{RF} = {}^{O}I_c\,{}^{O}\boldsymbol{\alpha}_C $$

$$ {}^{O}\boldsymbol{F}_{\mathbf{net},B} = {}^{O}\boldsymbol{F}_{GB} + {}^{O}\boldsymbol{F}_{T} + {}^{O}\boldsymbol{F}_{CB} = m_B\,{}^{O}\boldsymbol{a}_B $$
$$ {}^{O}\boldsymbol{\tau}_{\mathbf{net},B} = R(\boldsymbol{q}_B)\,{}^{B}\boldsymbol{\tau}_{CB} = {}^{O}I_B\,{}^{O}\boldsymbol{\alpha}_B $$

System State

- Define state of the system
- $\boldsymbol{p}_{sys} = \boldsymbol{p}_B$ and $\boldsymbol{q}_{sys} = \boldsymbol{q}_B$,

$$\begin{bmatrix} \boldsymbol{p}_C \\ \boldsymbol{q}_C \end{bmatrix} = \begin{bmatrix} \boldsymbol{p}_B + \boldsymbol{r}_{BC} \\ \boldsymbol{q}_\theta \boldsymbol{q}_B \end{bmatrix} = \begin{bmatrix} \boldsymbol{p}_{sys} + \boldsymbol{r}_{BC} \\ \boldsymbol{q}_\theta \boldsymbol{q}_{sys} \end{bmatrix}$$

$$\begin{bmatrix} \dot{\boldsymbol{p}}_C \\ \dot{\boldsymbol{q}}_C \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{p}}_{sys} + \dot{R}(\boldsymbol{q}_{sys})\, {}^B\boldsymbol{r}_{BC} \\ \boldsymbol{q}_\theta \dot{\boldsymbol{q}}_{sys} + \dot{\boldsymbol{q}}_\theta \boldsymbol{q}_{sys} \end{bmatrix}$$

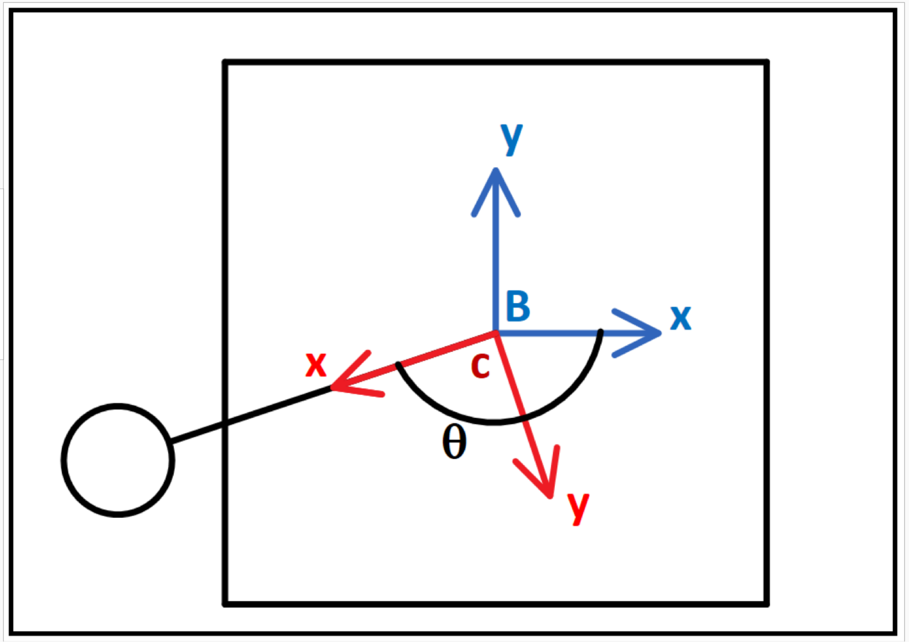
$$\begin{bmatrix} \ddot{\boldsymbol{p}}_C \\ \ddot{\boldsymbol{q}}_C \end{bmatrix} = \begin{bmatrix} \ddot{\boldsymbol{p}}_{sys} + \ddot{R}(\boldsymbol{q}_{sys})\, {}^B\boldsymbol{r}_{BC} \\ \boldsymbol{q}_\theta \ddot{\boldsymbol{q}}_{sys} + 2[\dot{\boldsymbol{q}}_\theta \dot{\boldsymbol{q}}_{sys}] + \ddot{\boldsymbol{q}}_\theta \boldsymbol{q}_{sys} \end{bmatrix}$$

State Variable: θ

- Yaw angle difference
- Turn into Quaternion expression

$$\boldsymbol{q_\theta} = \cos(\frac{\theta}{2}) + \sin(\frac{\theta}{2}) R(\boldsymbol{q_{sys}})\, {}^{B}\hat{\boldsymbol{z}}_{\boldsymbol{B}}$$

$$\dot{\boldsymbol{q}}_{\boldsymbol{\theta}} = -\frac{1}{2}\sin(\frac{\theta}{2})\dot{\theta} + \frac{1}{2}\cos(\frac{\theta}{2})\dot{\theta} R(\boldsymbol{q_{sys}})\, {}^{B}\hat{\boldsymbol{z}}_{\boldsymbol{B}} + \sin(\frac{\theta}{2}) R(\dot{\boldsymbol{q}}_{\boldsymbol{sys}})\, {}^{B}\hat{\boldsymbol{z}}_{\boldsymbol{B}}$$



$$\zeta = 2I_B(\dot{\boldsymbol{q}}_{sys}\boldsymbol{q}^*_{sys})^2 + 2I_C[(\boldsymbol{q}_0\dot{\boldsymbol{q}}_{sys} + \dot{\boldsymbol{q}}_0\boldsymbol{q}_{sys})(\boldsymbol{q}_0\boldsymbol{q}_{sys})^*]^2 - 4I_C(\dot{\boldsymbol{q}}_0\dot{\boldsymbol{q}}_{sys})(\boldsymbol{q}_0\boldsymbol{q}_{sys})^*$$

$$\boldsymbol{F}_{BC} = m_B\ddot{\boldsymbol{p}}_{sys} - \boldsymbol{F}_{GB} - \boldsymbol{F}_T$$

System of equations

$$(m_b + m_c)\ddot{\boldsymbol{p}}_{sys} + m_c\ddot{R}(\boldsymbol{q}_{sys})\,{}^B\boldsymbol{r}_{BC} = \boldsymbol{F}_{GC} + \boldsymbol{F}_{GB} + \boldsymbol{F}_T$$

$$2I_B[\ddot{\boldsymbol{q}}_{sys}\boldsymbol{q}^*_{sys}] + 2I_C[\boldsymbol{q}_0\ddot{\boldsymbol{q}}_{sys}(\boldsymbol{q}_0\boldsymbol{q}_{sys})^*] + 2I_C[\ddot{\boldsymbol{q}}_0\boldsymbol{q}_{sys}](\boldsymbol{q}_0\boldsymbol{q}_{sys})^* - \boldsymbol{r}_{CB} \times \boldsymbol{F}_{BC} = \zeta$$

$$q_r\ddot{q}_r + q_i\ddot{q}_i + q_j\ddot{q}_j + q_k\ddot{q}_k + \dot{q}_r^{\,2} + \dot{q}_i^{\,2} + \dot{q}_j^{\,2} + \dot{q}_k^{\,2} = 0$$

- 8 equations, 8 unknows, $f(\ddot{\boldsymbol{p}}, \ddot{\boldsymbol{q}}, \ddot{\theta}, \dot{\boldsymbol{p}}, \dot{\boldsymbol{q}}, \dot{\theta}, \boldsymbol{p}, \boldsymbol{q}, \theta) = 0$
- Should be able to

$$\text{solve for } \ddot{\boldsymbol{p}}, \ddot{\boldsymbol{q}}, \ddot{\theta} \text{ given } \dot{\boldsymbol{p}}, \dot{\boldsymbol{q}}, \dot{\theta}, \boldsymbol{p}, \boldsymbol{q}, \theta.$$

Matlab Implementation

- State evolution equation
- $s_{t+1} = s_t + \dot{s}_t \Delta t$

$$\boldsymbol{s_{sys}} = \begin{bmatrix} \dot{\boldsymbol{p}} \\ \dot{\boldsymbol{q}} \\ \dot{\theta} \\ \boldsymbol{p} \\ \boldsymbol{q} \\ \theta \end{bmatrix} \quad \text{so that} \quad \dot{\boldsymbol{s}}_{\boldsymbol{sys}} = \begin{bmatrix} \ddot{\boldsymbol{p}} \\ \ddot{\boldsymbol{q}} \\ \ddot{\theta} \\ \dot{\boldsymbol{p}} \\ \dot{\boldsymbol{q}} \\ \dot{\theta} \end{bmatrix}$$

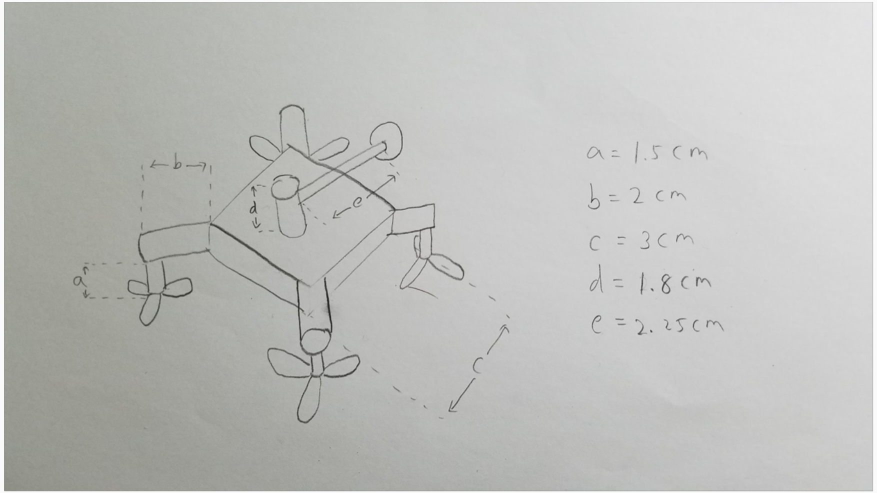
- Unfortunately, implementation in Matlab yield no solutions....

Math Modelling: Challenges

- Rotation is hard
- No close form solution
- Rely on numerical method
- A lot can go wrong

Simulation: Building 3D Geometric Structure

Starting from measuring and sketching



Making a list of geometric components

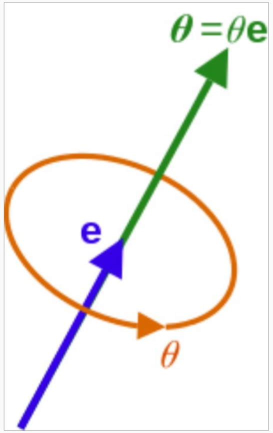
Main body -- Box shape: 1
Motor -- Cylinder shape: 1
Mass stick -- Cylinder shape: 1
Mass -- Sphere shape: 1
Leg -- Cylinder shape: 4
Propeller holder -- Cylinder shape:4
Propeller : 4

Simulation: Building 3D Geometric Structure

Simulink: 3D world editor

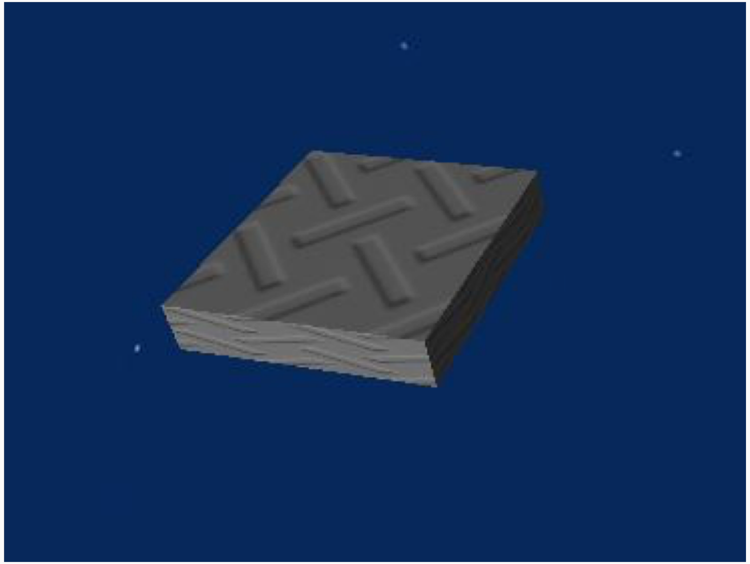
Axis-angle representation(Quaternion)

$$(\text{axis}, \text{angle}) = \left(\begin{bmatrix} e_x \\ e_y \\ e_z \end{bmatrix}, \theta \right) = \left(\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \frac{\pi}{2} \right)$$

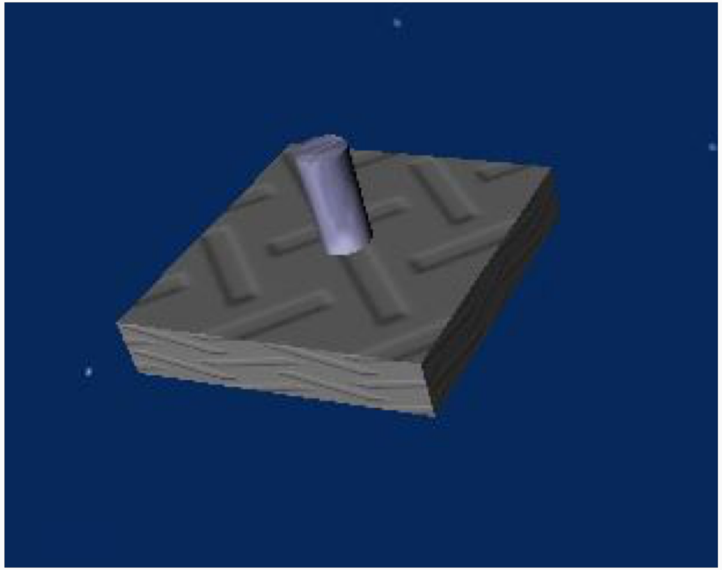


Simulation: Building 3D Geometric Structure

Starting with the main body

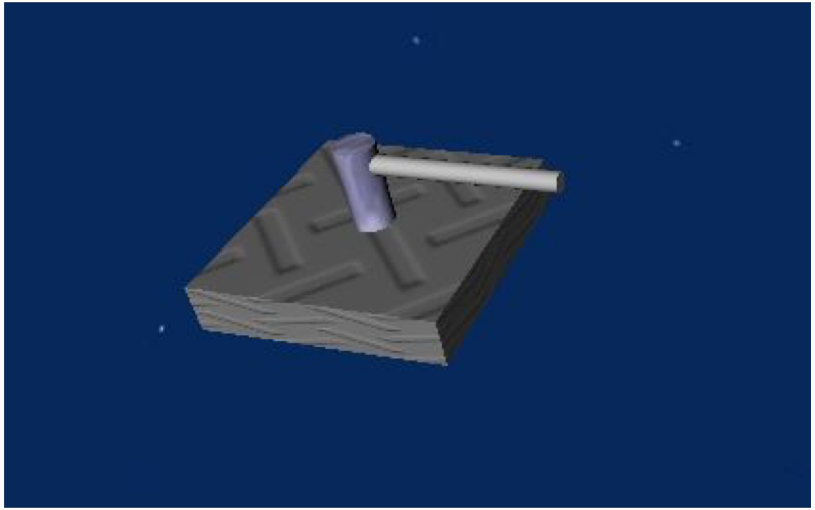


Adding the motor

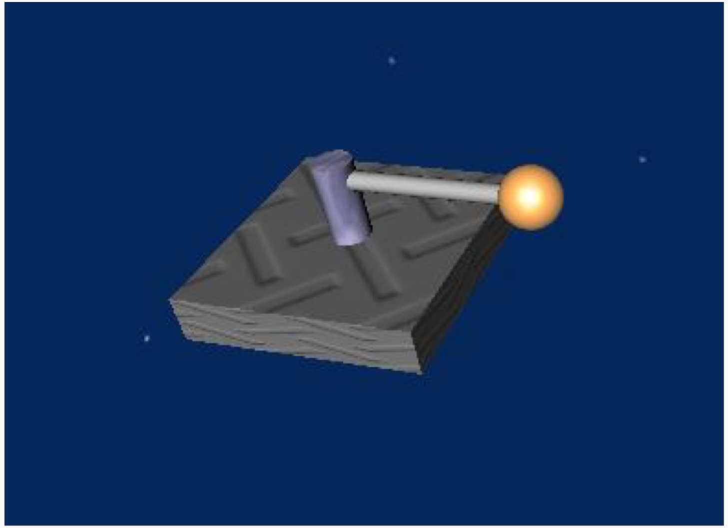


Simulation: Building 3D Geometric Structure

Adding the spinning stick

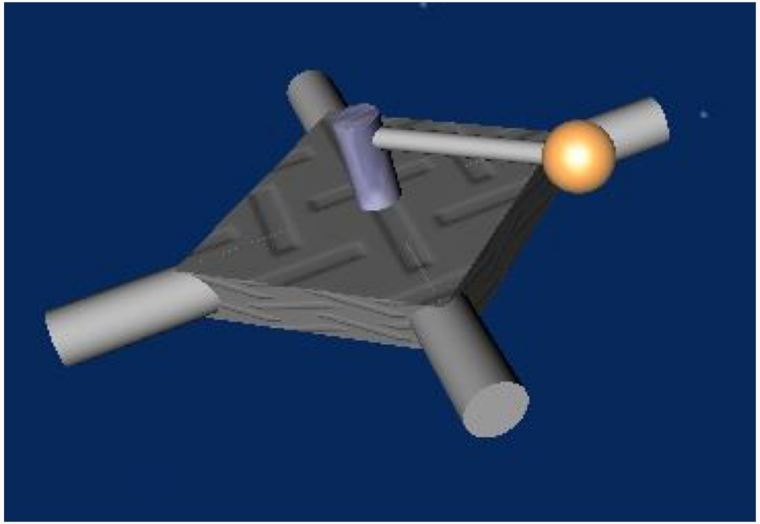


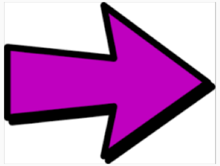
Adding the mass



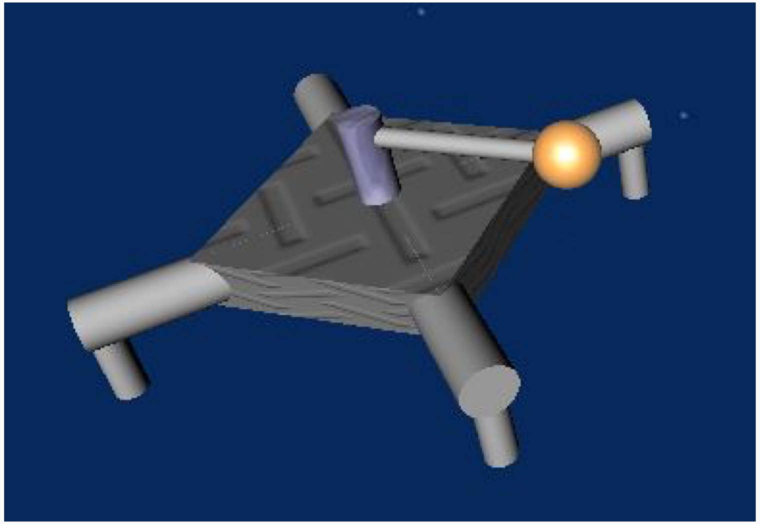
Simulation: Building 3D Geometric Structure

Adding the legs

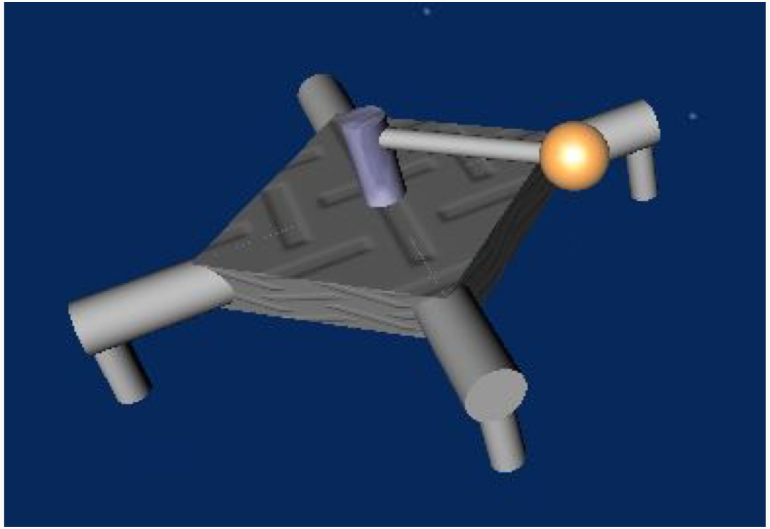


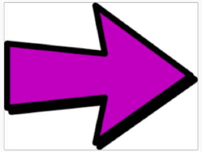


Adding the propeller holders

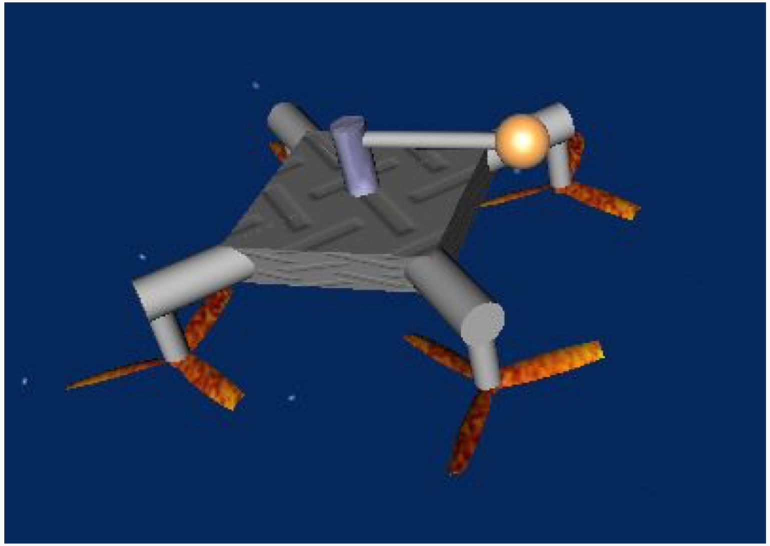


Simulation: Building 3D Geometric Structure



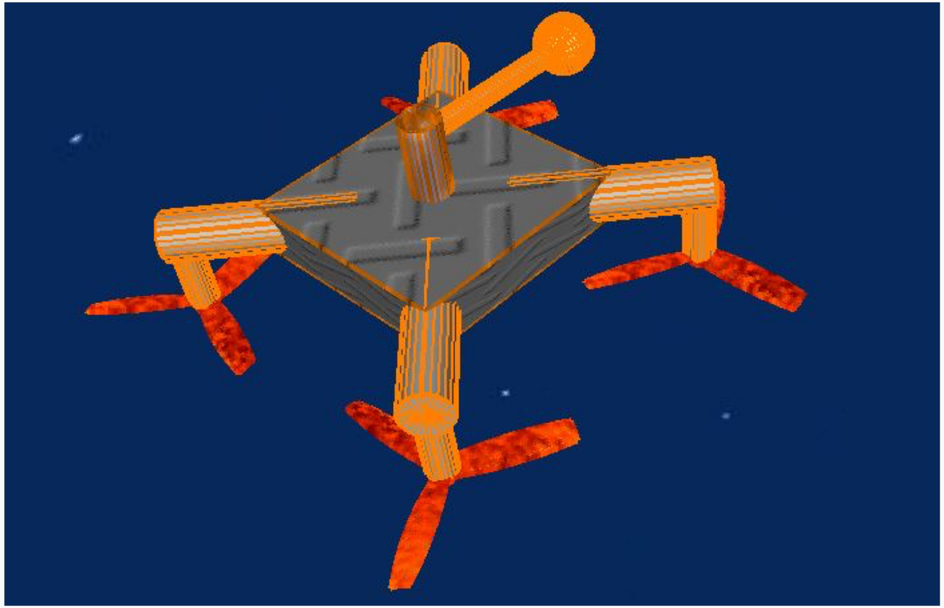


Adding the propellers



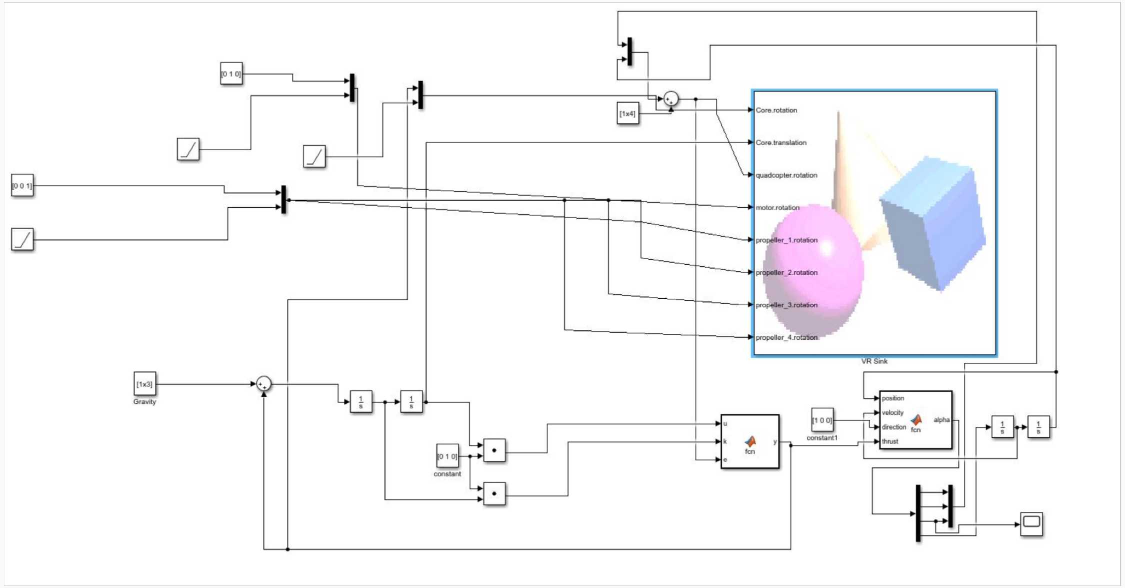
Simulation: Building 3D Geometric Structure

- Core (Transform)
  - center (SFVec3f): 0 0 0
  - rotation (SFRotation): 0 0 1 0
  - scale (SFVec3f): 1 1 1
  - scaleOrientation (SFRotation): 0 0 1 0
  - translation (SFVec3f): 0 0 0
  - bboxCenter (SFVec3f): 0 0 0
  - bboxSize (SFVec3f): -1 -1 -1
  - children (MFNode)
    - (Shape)
    - quadcopter (Transform)
      - center (SFVec3f): 0 0 0
      - rotation (SFRotation): 0 0 100 0
      - scale (SFVec3f): 1 1 1
      - scaleOrientation (SFRotation): 0 0 1 0
      - translation (SFVec3f): 0 0 0
      - bboxCenter (SFVec3f): 0 0 0
      - bboxSize (SFVec3f): -1 -1 -1
      - children (MFNode)
        - (Shape)
        - motor (Transform)
        - (Shape)
        - Leg1 (Transform)
        - Leg2 (Transform)
        - Leg3 (Transform)
        - Leg4 (Transform)



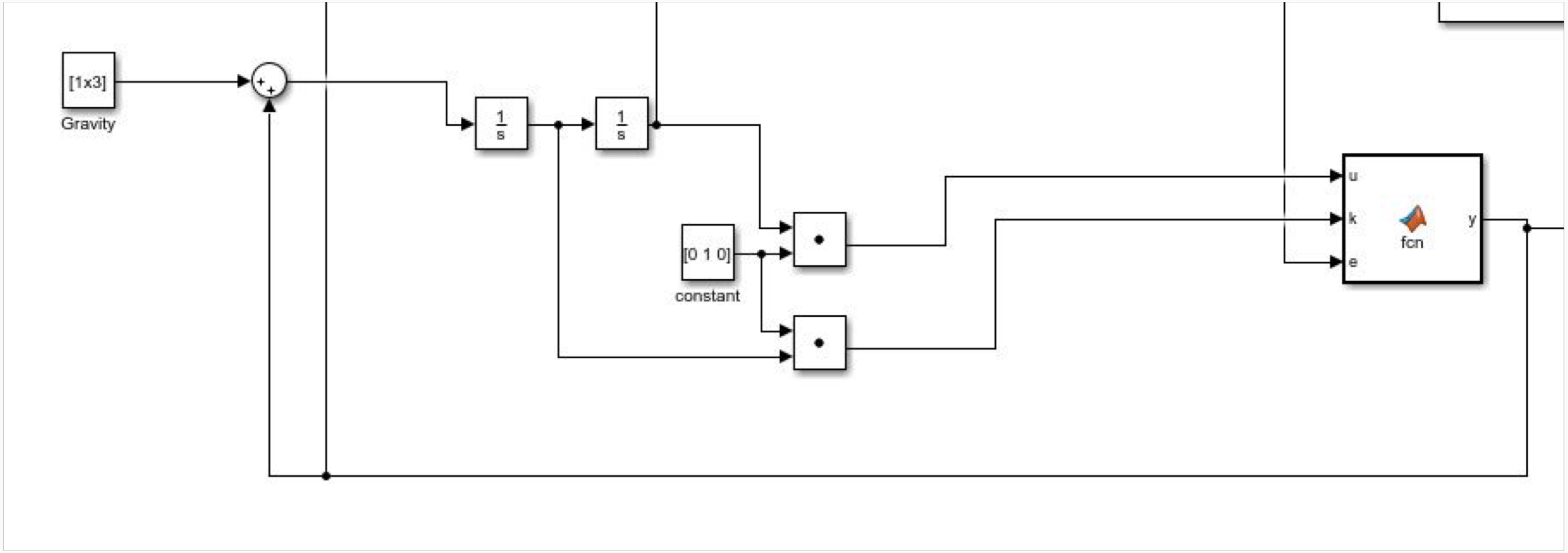
Simulation: Updating State Information

Simulink block diagram



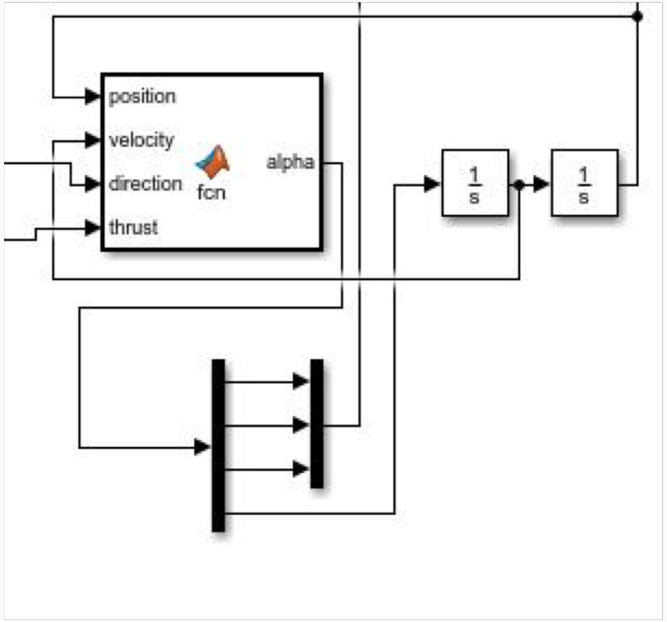
Simulation: Updating State Information

The net force calculating system



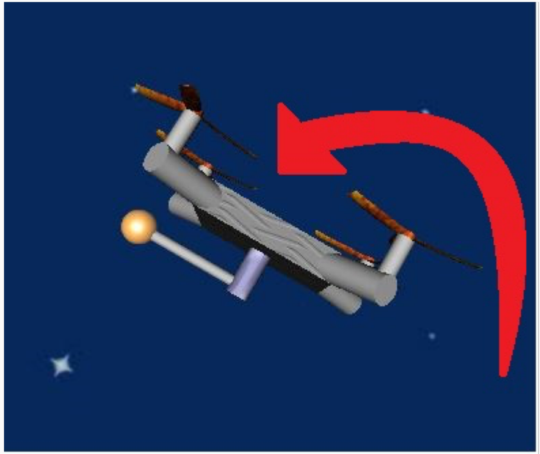
Simulation: Updating State Information

The net torque calculating system

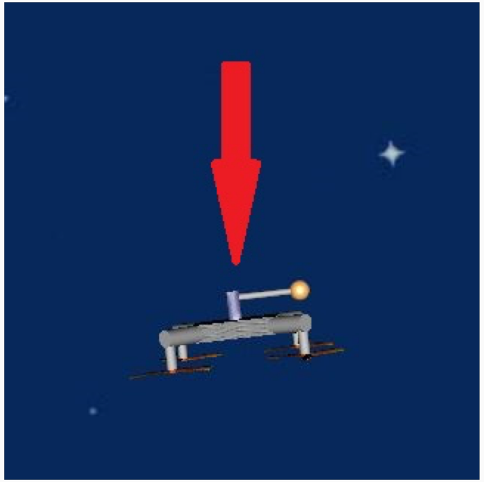


Simulation: Conclusion

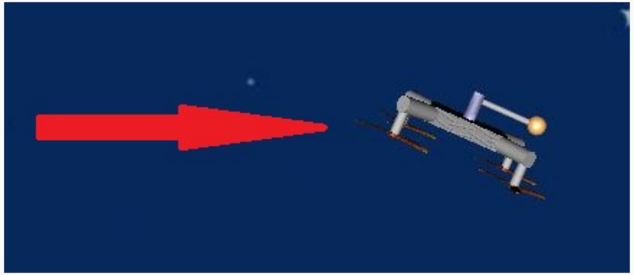
Low spinning rate



High load/max load ratio



Optimal parameter setting



Simulation Demo Video: https://www.youtube.com/watch?v=o9f2x5YUPoA

Simulation: Challenges

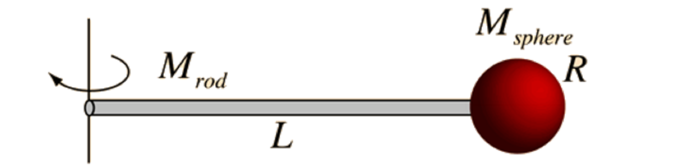
- Getting used to quaternion representation
- Making assumptions
- Adding more and more math details

Conclusion: Model and Simulation

- Hard control problem
- Deriving an exact mathematical model may be even harder
- Spent 5 weeks on this
- Focus started to shift towards the practical implementation

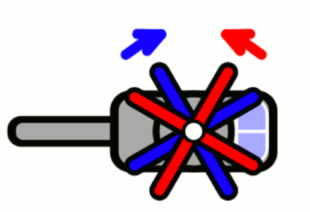
Model Difficulties

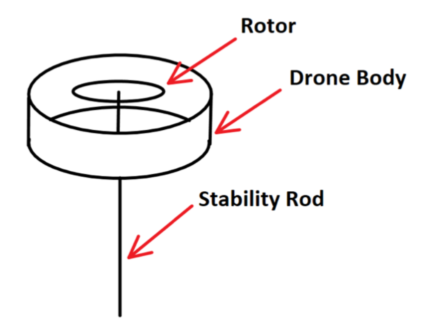
- Vijay Kumar
- Math becoming increasingly difficult
- Redesign & coaxial rotor design
- Counter-Torque spin of mass
- Refocus on mass actuator

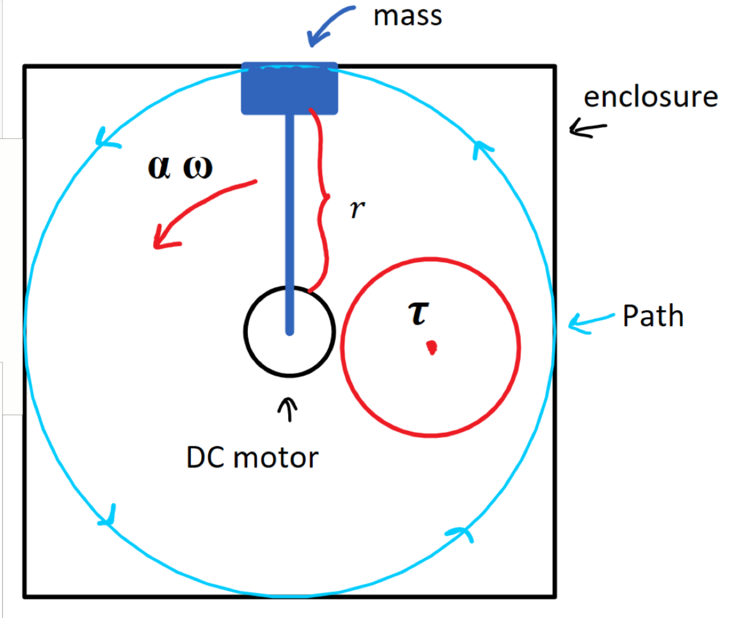


$$I = \frac{1}{3} M_{rod} L^2 + \frac{2}{5} M_{sphere} R^2 + M_{sphere} (L+R)^2$$

$$I = I_{\text{rod about end}} + I_{\text{sphere about center}} + \text{Parallel axis contribution}$$







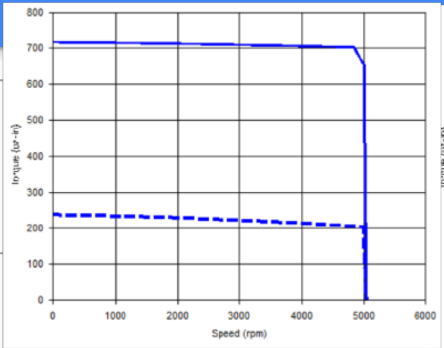
Motor Comparison: Torque & Speed

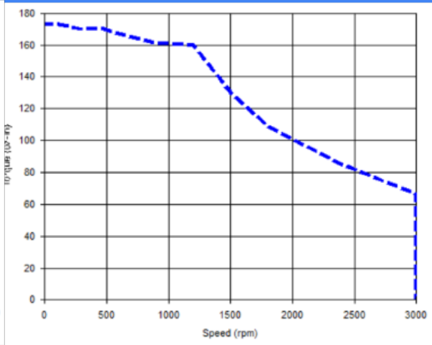
Motor	**Pros**	**Cons**
Stepper	lower speed, Torque control, encoder	Weight, size, energy
Servo	High speed & torque, energy efficient, weight	Lower speed range
Permanent Magnet DC	Great Starting torque, good speed regulation	Limited Torque
Series DC	Large Starting Torque	No speed regulation
Shunt DC	Great speed regulation	Low Starting Torque
Compound	Good Starting Torque	Poor Speed Regulation

UCLA ENGINEERING

Motor Conclusion

Stepper w/ Enc.	Servo w/ Enc.	Permanent Magnet DC w/ Enc.
Stall Prevention Stall Detection Torque Control No tuning when commanding position Heat due to full current draw Lose torque as speed is increased	Increase current/torque to correct for errors in motor speed Require tuning when commanding position Flat Torque vs speed curve	Absolute encoder allows the determination of position. Can use PID regulator Can calculate speed from angular position





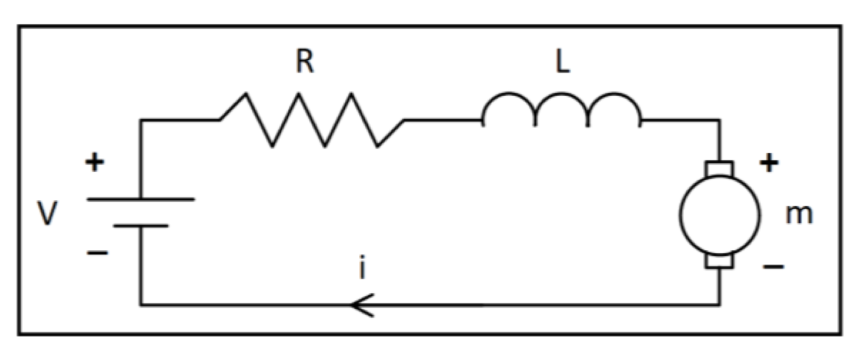
- CrazyFlie Limitations
- 7mm brushed DC
- Weight: 2.7g
- Kv: 14000 rpm/V
- Medium sized drone
- Stepper or Servo

Motor Control: Physical Representation

Assumptions:

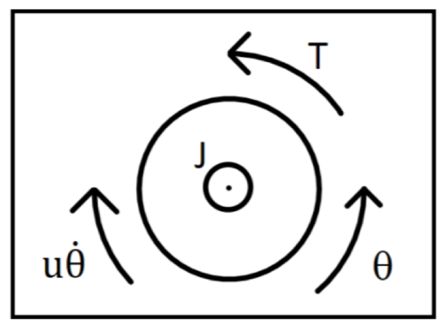
$$T = K_i \psi i$$

- Voltage input
- Rotational speed output
- Rigid components
- Constant E-Field
- Friction Torque is proportional to angular velocity

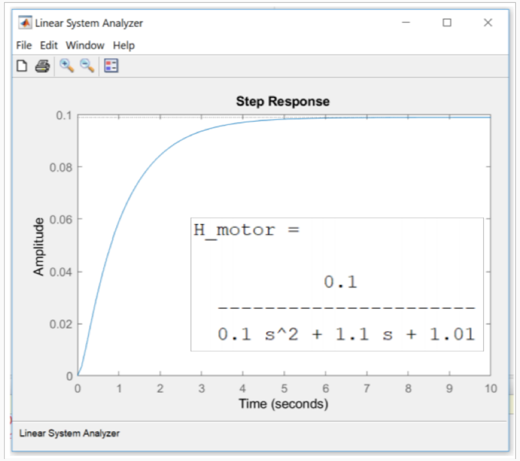


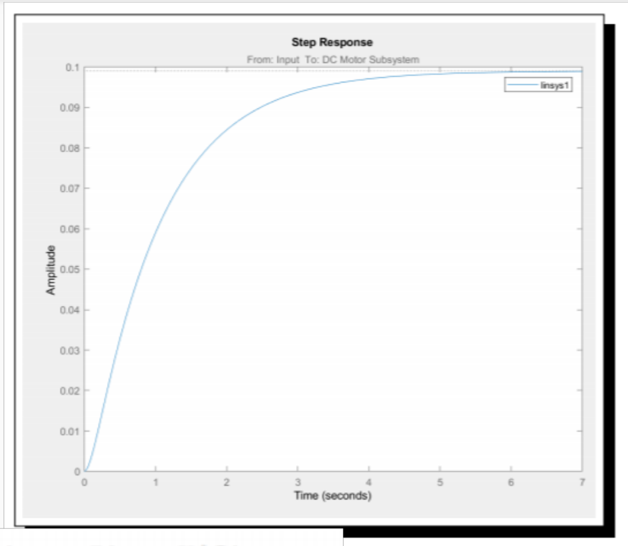
$$L\frac{di}{dt} + Ri + K_i\dot{\theta} = V$$

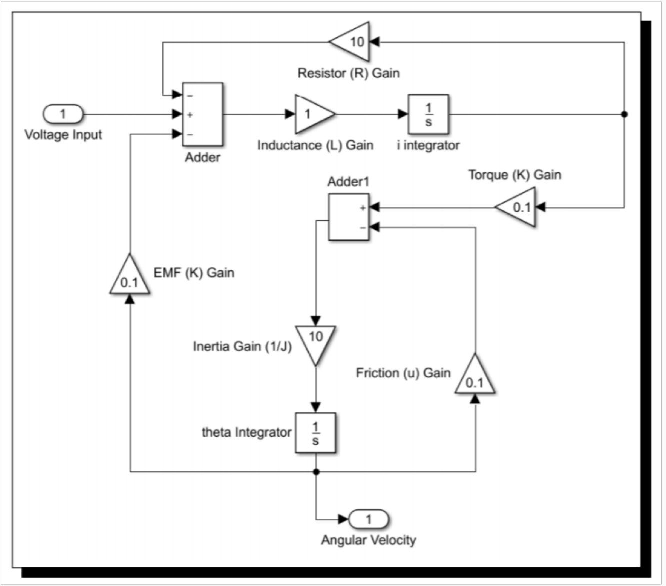
$$J\ddot{\theta} + u\dot{\theta} = K_i i$$



Motor Control: Speed



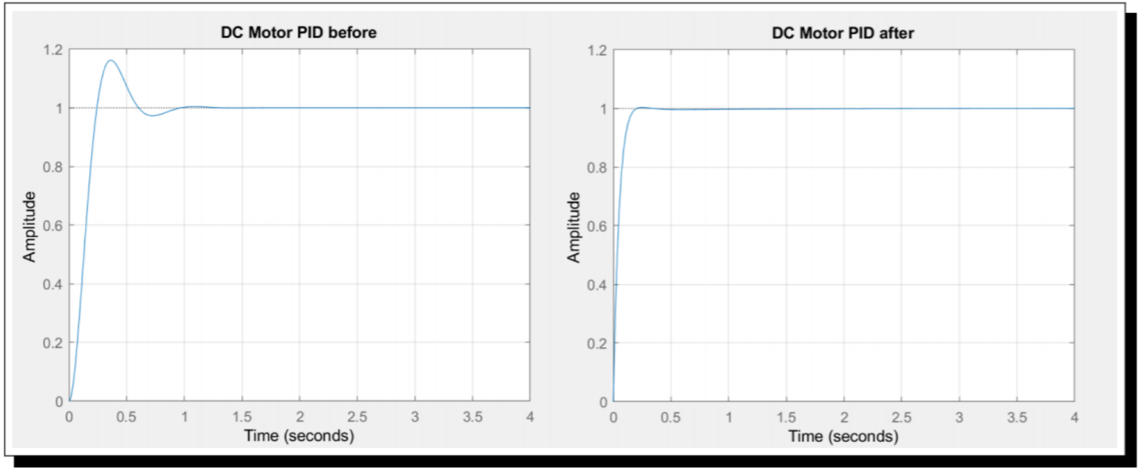


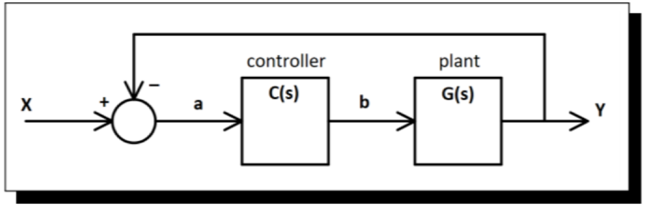



```
H_motor = K / ((J*s + u ) * (L*s + R) + K^2);
display(H_motor);

linearSystemAnalyzer('step', H_motor, 0:0.1:10);
```

Motor Control: Speed





$$C(s) = K_p + \frac{K_i}{s} + K_d s$$

$$\frac{d}{dt}\begin{bmatrix}\dot{\theta} \\ i\end{bmatrix} = \begin{bmatrix} -\frac{u}{J} & \frac{K_i}{J} \\ -\frac{K_i}{L} & -\frac{R}{L}\end{bmatrix}\begin{bmatrix}\dot{\theta} \\ i\end{bmatrix} + \begin{bmatrix}0 \\ L^{-1}\end{bmatrix} V$$

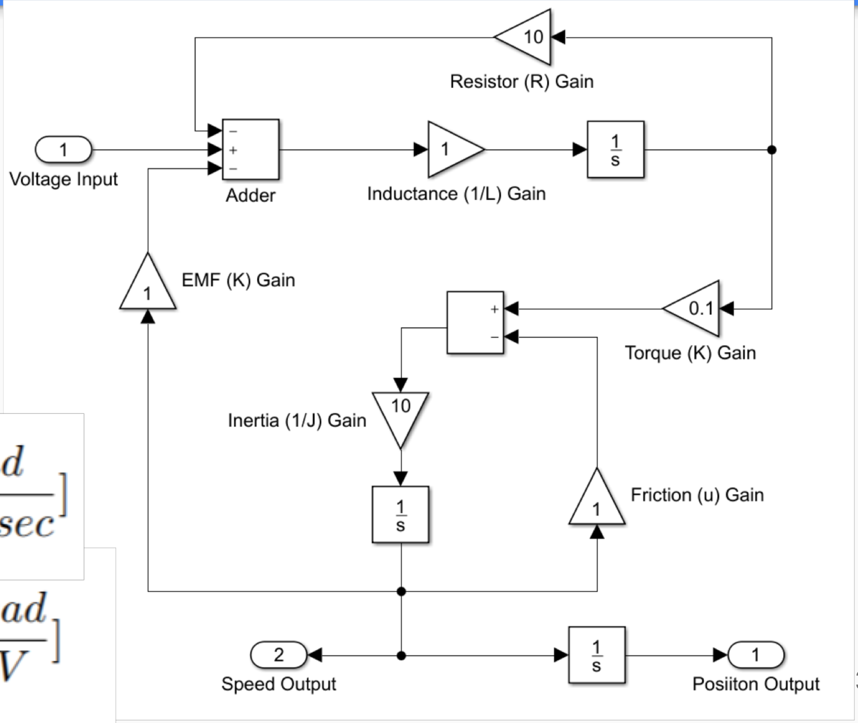
$$z = \begin{bmatrix}0 & 1\end{bmatrix}\begin{bmatrix}\dot{\theta} \\ i\end{bmatrix}$$

Motor Control: Position

- Simulink Simscape
- ssc_dcmotor
- Similar assumptions
- DC Values need to be measured in lab

$$H(s) = \frac{\dot{\Theta}(s)}{V(s)} = \frac{K_i}{(Js+u)(Ls+R)+K_i^2} \qquad [\frac{rad}{V \cdot sec}]$$

$$H(s) = \frac{\Theta(s)}{V(s)} = \frac{K_i}{s((Js+u)(Ls+R)+K_i^2)} \qquad [\frac{rad}{V}]$$



Challenges

- Mathematical modeling
- OCSM Dynamics can be tricky
- Modeling various types of motors is time consuming
- Getting Motors we can test
- Cost of the motors we need

UCLA ENGINEERING

Actual Implementation

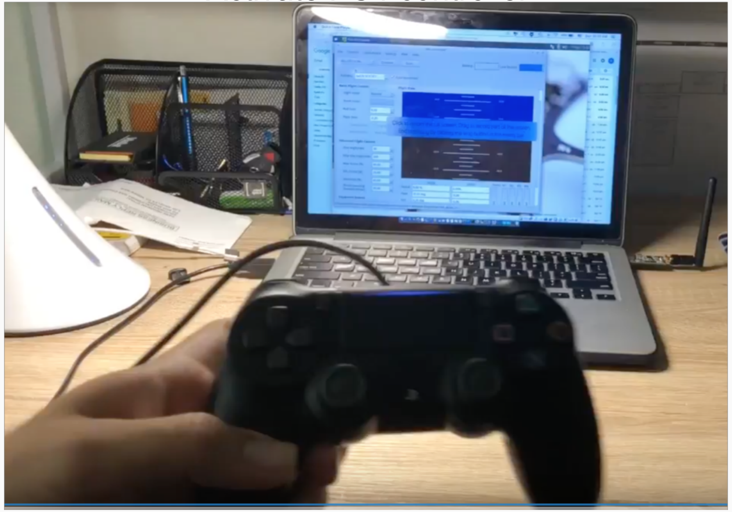
1. CF2.0 Software resources /coding:

Getting sensor measurements from CF2.0 IMU

Testing

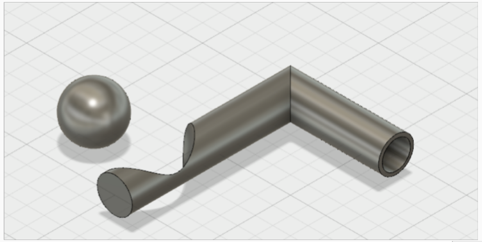
```
In [4]: %run ../examples/basiclogSync.py
[35500][<cflib.crazyflie.log.LogConfig object at 0x10d03f128>]: {'stabilizer.roll': 1.5027883052825928, 'stabilizer.p
itch': -1.303802728652954, 'stabilizer.yaw': 0.13622625172138214}
[35510][<cflib.crazyflie.log.LogConfig object at 0x10d03f128>]: {'stabilizer.roll': 1.5025050640106201, 'stabilizer.p
itch': -1.3042619228363037, 'stabilizer.yaw': 0.1369791179895401}
[35520][<cflib.crazyflie.log.LogConfig object at 0x10d03f128>]: {'stabilizer.roll': 1.5026928186416626, 'stabilizer.p
itch': -1.3046281337738037, 'stabilizer.yaw': 0.13777440786361694}
[35530][<cflib.crazyflie.log.LogConfig object at 0x10d03f128>]: {'stabilizer.roll': 1.502584934234619 1, 'stabilizer.p
itch': -1.304977536201477, 'stabilizer.yaw': 0.13840070366859436}
[35540][<cflib.crazyflie.log.LogConfig object at 0x10d03f128>]: {'stabilizer.roll': 1.50305306911468 5, 'stabilizer.pi
tch': -1.304872751235962, 'stabilizer.yaw': 0.13785721361637115}
[35550][<cflib.crazyflie.log.LogConfig object at 0x10d03f128>]: {'stabilizer.roll': 1.502662181854248, 'stabilizer.pi
tch': -1.3051222562789917, 'stabilizer.yaw': 0.13770698010921478}
[35560][<cflib.crazyflie.log.LogConfig object at 0x10d03f128>]: {'stabilizer.roll': 1.5024116039276123, 'stabilizer.p
itch': -1.3050979375839233, 'stabilizer.yaw': 0.1381182074546814}
[35570][<cflib.crazyflie.log.LogConfig object at 0x10d03f128>]: {'stabilizer.roll': 1.502823829650879, 'stabilizer.pi
tch': -1.305249571800232, 'stabilizer.yaw': 0.13809622824192047}
[35580][<cflib.crazyflie.log.LogConfig object at 0x10d03f128>]: {'stabilizer.roll': 1.5024672746658325, 'stabilizer.p
itch': -1.3054291009902954, 'stabilizer.yaw': 0.13772787153720856}
[35590][<cflib.crazyflie.log.LogConfig object at 0x10d03f128>]: {'stabilizer.roll': 1.5030161142349243, 'stabilizer.p
itch': -1.3046841621398926, 'stabilizer.yaw': 0.1377219557762146}
```

Activate PS4 controller

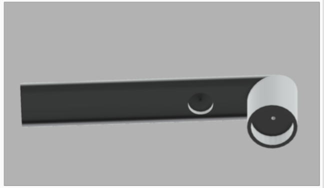


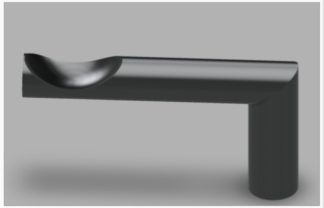
Actual Implementation

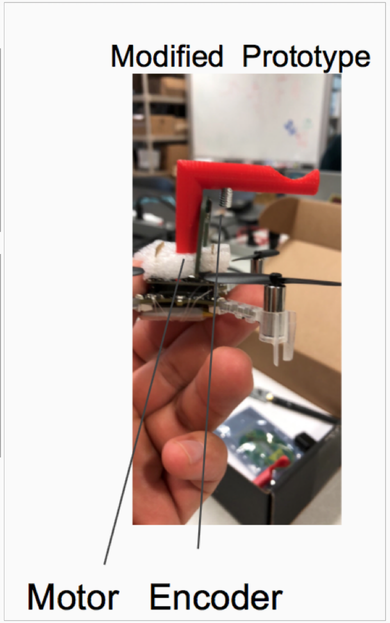
2. Rotating arm design:



Modifications added







Max Payload	15 g
Motor weight	2.7g
Spinning mass	<10g

Actual Implementation

3. Choose the DC motor to use :

Max Payload	15 g
Motor weight	2.7g
Spinning mass	<10g

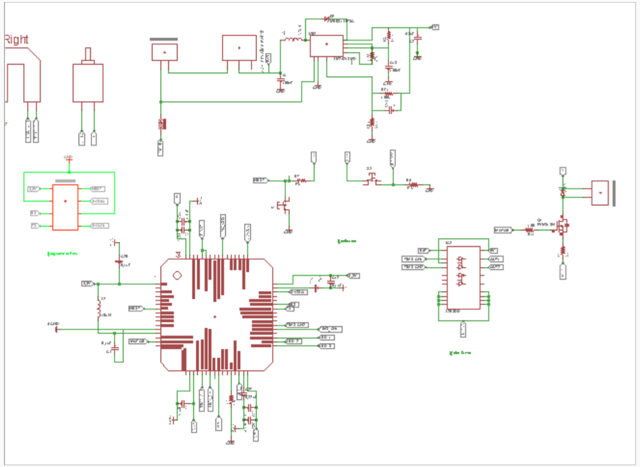
Use the same motor as CF2's motors:

1. Mainly due the payload limit
2. Current and voltage ratings matched with our battery constraints
    a. Rated voltage 4.2V
    b. Rated current 1000mA
    c. Test results in lab:
        i. Run at V=0.5v, I=0.4A
        ii. Stall current~ 3.5A
        iii. Handle up to 40g mass

Actual Implementation

4. Circuit Design :

Initially began with designing schematics for PCB:



Needed Components:

1. MCU
2. Motor Driver
3. Voltage Regulator
4. DC motor
5. Encoder
6. Programmer pins
7. RF/Bluetooth Module

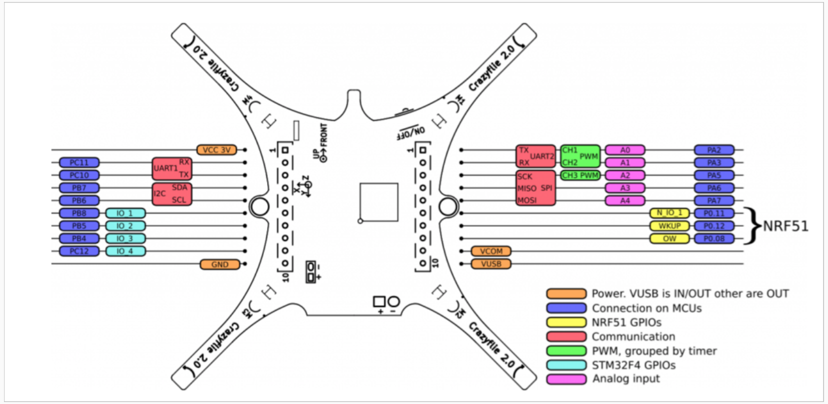
a. Same MCU as CF
b. Voltage regulator is not needed
c. RF/Bluetooth module+programming (through micro usb is provided)
d. More robust connection to user controller

Actual Implementation

4. Circuit Design :

So decided to use the I/O pins on CF2 MCU:

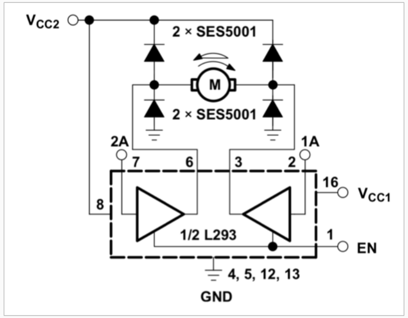
1. IO 1: for PWM pin
2. GND
3. VCC (3V)
4. A1 for Encoder

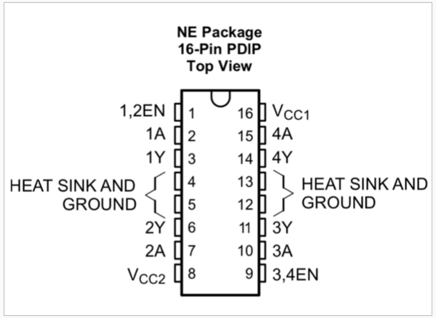


Actual Implementation

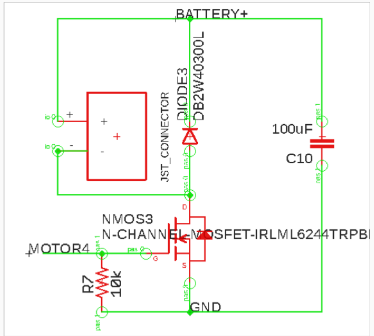
5. Motor Driver :

A. Bidirectional motor driver using H-bridge:





B. Unidirectional motor driver:

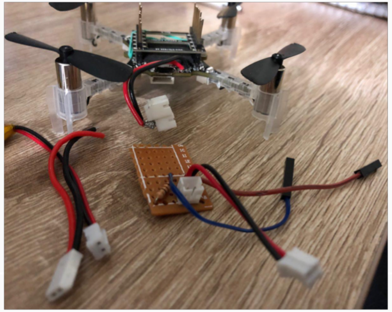


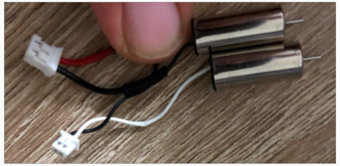
Actual Implementation

5. Motor Driver :

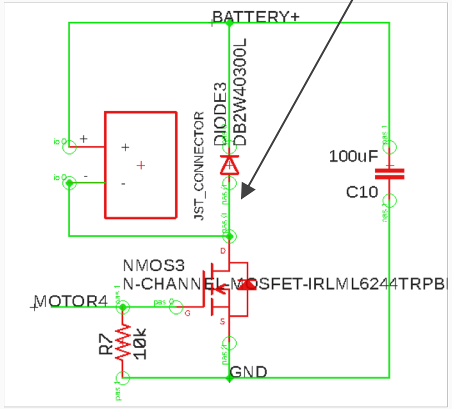
Both designed were tested on CF2's MCU, however uni directional design was picked to be soldered on perfboard for implementation on MCU mainly for simplicity:

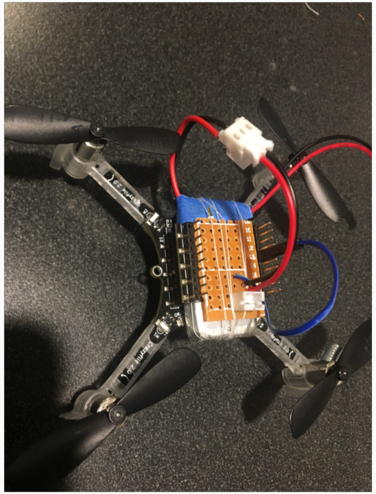
Additional connections with different JST connector were used to build the circuit on perfboard





10 ohm Resistor added here





Demo Link : https://youtu.be/xGmaOrivyys

Actual Implementation Challenges

- Crazyflie interface
  - Hard to code
    - PWM
- Quadcopter Dynamics is hard
- Motor driver
- Planning

Conclusion

- Researchers can look into our work if they have similar ideas as ours, we explored different possibilities on this hard control problem
- Plan better and execute the plans wiser,
  - Ex: started hacking the quadcopter in the first 5 weeks,
  - Instead of waiting for the math model to come out until 5th week
- Underactuated Robotics is very math based, since it makes use of system's dynamics

Expectation of live demo

- Show our work in each domain
- Paper that shows our Math model -- Wilson
- Computer that we can play with the simulation -- Lin
- Motor research and possibly a simulation -- Angel
- Actual implementation of the quadcopter -- Amir

References

https://www.bitcraze.io/2014/08/crazyflie-2-0-expansion-port/
https://store.bitcraze.io/collections/spare-parts/products/7-mm-dc-motor \break
https://www.micromo.com/technical-library/dc-motor-tutorials/motor-calculations
https://en.wikipedia.org/wiki/Armature_Controlled_DC_Motor
http://tutorial.math.lamar.edu/pdf/Laplace_Table.pdf
https://en.wikipedia.org/wiki/State-space_representation
http://ctms.engin.umich.edu/CTMS/index.php?aux=Home
http://www.ecircuitcenter.com/Circuits/dc_motor_model/DCmotor_model.htm