

CS591

UAV or Drone Simulation

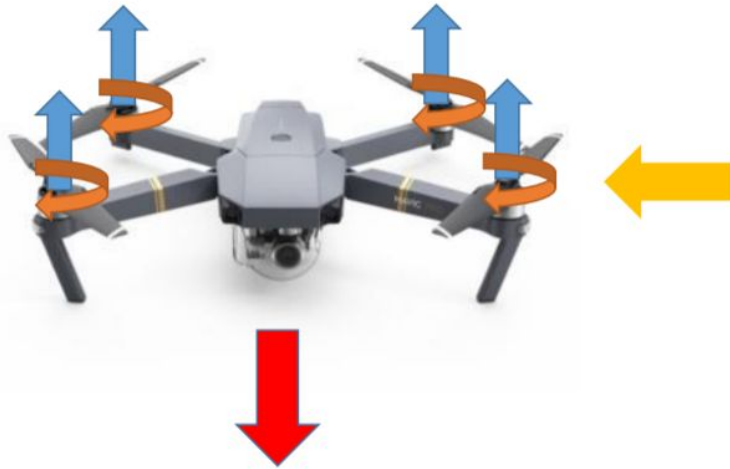
Samay Varshney
180101097

Introduction - UAV or Drone

- Four propeller drone modelling system
- Drone Characteristics used:
 - Diagonal length: 335mm
 - Battery Capacity: 3830mAh
 - Weight: 0.743kg
 - Voltage: 11.4V
 - Fontal Area: 0.0197m²
 - Max Discharge Current: 77A
 - Motor KV: 1400 rpm/V
 - Propeller Size: 8x4 inches



Forces at Play - UAV or Drone



↑ : Motors' thrust

← : Drag and Gusts

↓ : Gravity

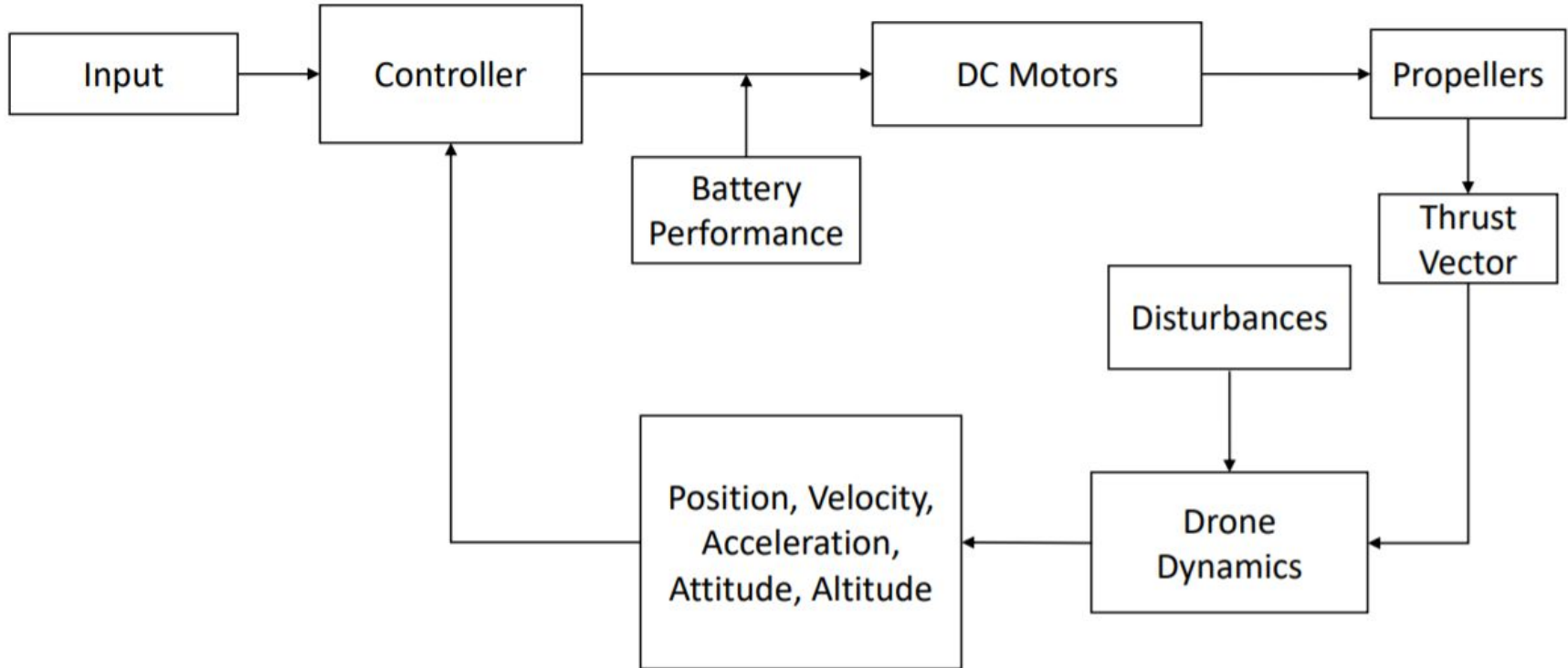
↻ : Motors/Propellers
Torque

Motor Limits - UAV or Drone

- Max operating voltage will be 11.4V to consider an average over one battery discharge and a max current of 77A.
- These characteristics will allow to limit the performance of our motors to realistic values by limiting the energy they are allowed to extract from the battery.
- DC motors are considered in this drone model to compute all the formulas.

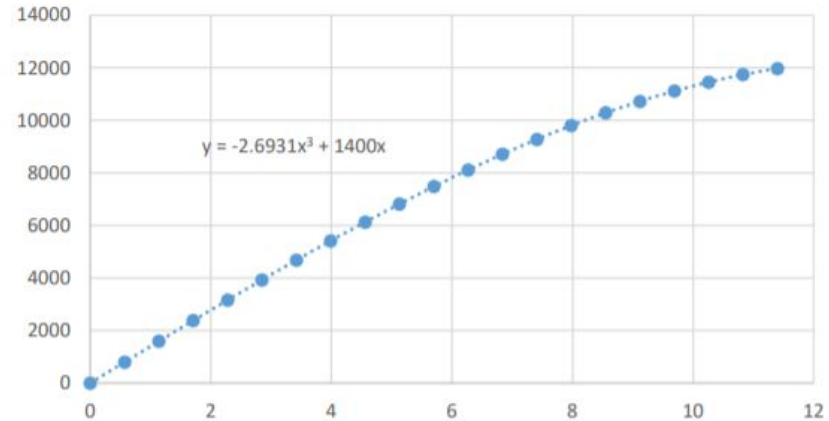


Simplified Model Diagram



RPM Motor Constant KV

- The APC, a company which manufactures propellers has published the detailed performance of its propellers on its website:
<https://www.apcprop.com/technical-information/performance-data/>
https://www.apcprop.com/files/PER3_8x4.dat
- I used 8x4 propeller data values for extracting different formulas for this drone dynamics by fitting polynomial equations. One example of deriving RPM is shown below.
- It contains for different RPMs and forward speeds the advance ratio, efficiency, coefficient of thrust, coefficient of power, power, torque and thrust for different types of propeller.



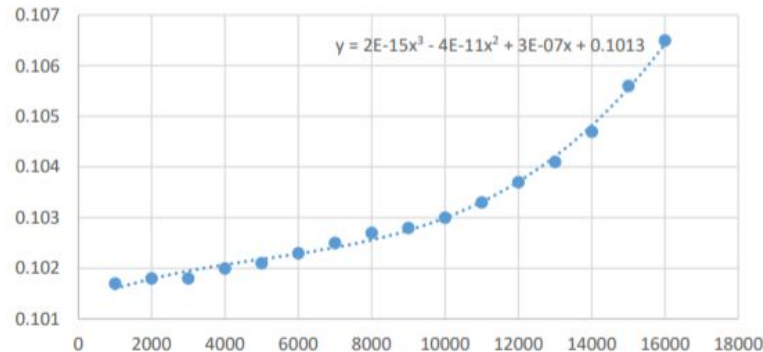
$$RPM = -2.6931 * V^3 + 1400 * V$$

Thrust Equation

- Thrust is obtained in Newtons from the coefficient of thrust using the following formula:

$$Thrust = C_T * \rho * n^2 * D^4$$

- C_T is the coefficient of thrust, ρ is the air density at sea level (1.225 kg/m³), n is the amount of revolutions per second and D is the diameter of the propeller in meters.
- Thrust will act perpendicular to the drone always.



$$C_t = 2 * 10^{-15} * RPM^3 - 4 * 10^{-11} * RPM^2 + 3 * 10^{-7} * RPM + 0.1013$$

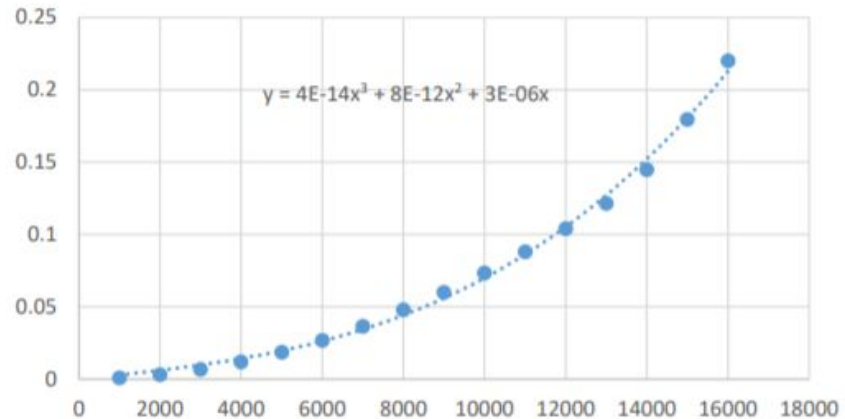
Torque Equation

- Propellers will load the motor with the torque required to overcome drag at the current RPM.
- In a DC motor, torque and current are proportional to each other.
- K_T is the Torque Constant in Nm/Amps and I is the current in Amps.
- The Torque Constant of the motor is the inverse of the motor RPM constant K_V .

$$\text{Torque} = K_T * I$$

$$K_T = \frac{1}{K_V} = \frac{1}{1400} = 0.0007$$

$$I = \text{Torque} * 1400$$



$$\text{Torque} = 4 * 10^{-14} * RPM^3 + 8 * 10^{-12} * RPM^2 + 3 * 10^{-6} * RPM$$

Shape of Drone and Axes

In the x direction: $F_x = ma_x$

In the y direction: $F_y = ma_y$

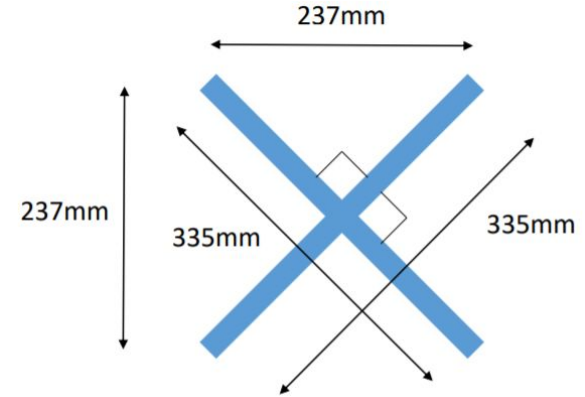
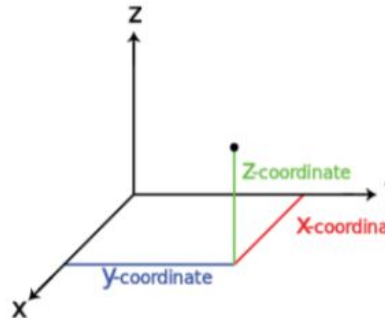
In the z direction: $F_z = ma_z$

About the x axis (pitch): $M_x = I_x * \ddot{\theta}_x$

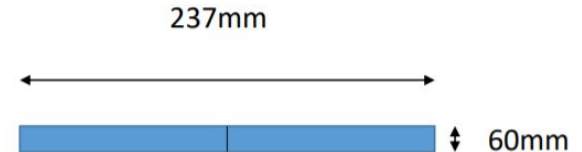
About the y axis (roll): $M_y = I_y * \ddot{\theta}_y$

About the z axis (yaw): $M_z = I_z * \ddot{\theta}_z$

M is the external moments or torques in Nm, I is the moments of inertia in kg.m^2 and $\ddot{\theta}$ is rotational acceleration in rad/s^2 .



Top View



Side View

Moment of Inertia

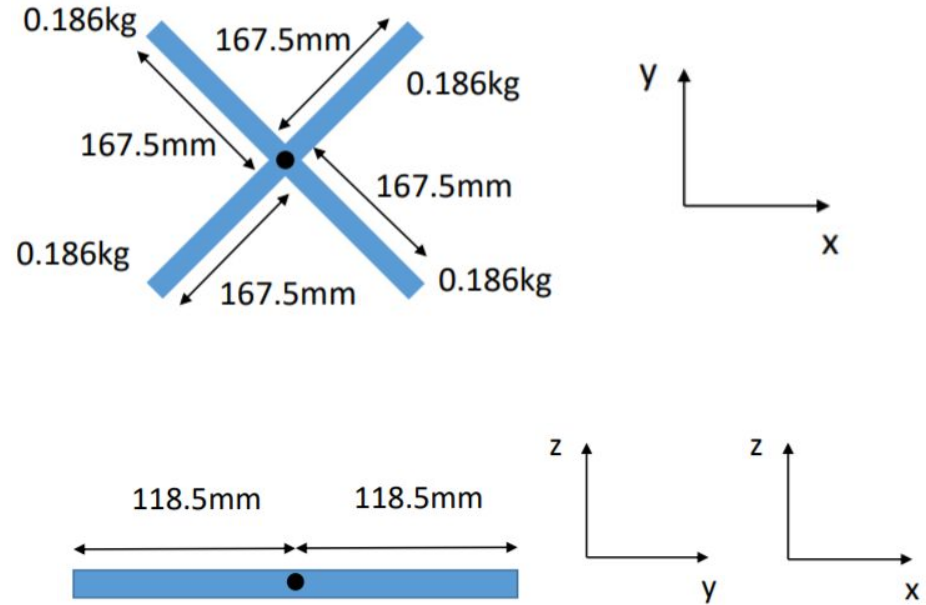
Moment of Inertia of a rod about its end:

$$I = \frac{1}{3}ML^2$$

$$I_z = \frac{1}{3} * \frac{0.743}{4} * \left(\frac{0.335}{2}\right)^2 * 4$$

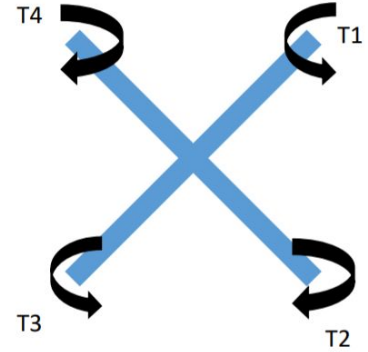
$$I_z = 0.007 \text{ kg.m}^2$$

$$I_{x,y} = \frac{1}{3} * \frac{0.743}{4} * \left(\frac{0.237}{2}\right)^2 * 4 = 0.003 \text{ kg.m}^2$$



Torques and Propellers Rotation

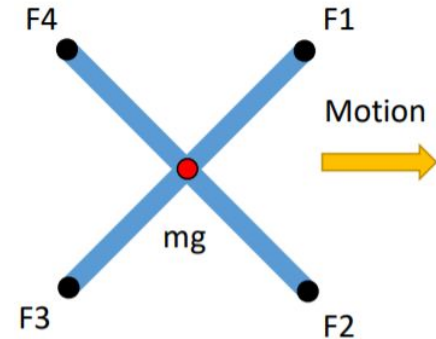
- The torques necessary to rotate each propellers also act on the drone.
- Props 2 and 4 rotate in the same direction and oppositely to 1 and 3.
- Decreasing or increasing the power in each of these couples independently induces yaw.



$$M_z = T4 - T1 + T2 - T3$$

Thrust and Moments

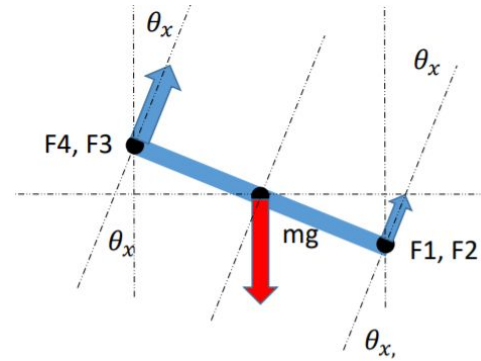
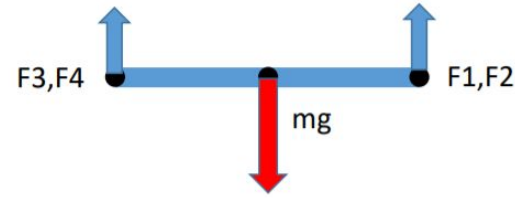
- F1, F2, F3 and F4 each represent the thrust of their respective propeller.
- F1 and F2 as well as F4 and F3 are respectively coupled to induce pitch.
- F4 and F1 as well as F3 and F2 are respectively coupled to induce roll.



Pitch and Roll Motions

- When pitching positively $F1$ and $F2$ decrease while $F3$ and $F4$ increase, leading the drone to pitch by θ_x . When rolling positively $F3$ and $F2$ increase while $F4$ and $F1$ decrease keeping total thrust constant but leading θ_y to vary.

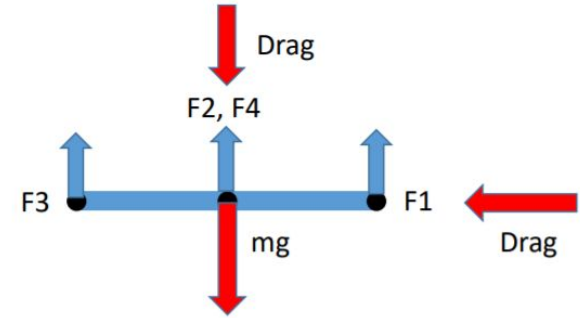
$$M_x = (F3 + F4) * \frac{0.237}{2} - (F1 + F2) * \frac{0.237}{2}$$
$$M_y = (F3 + F2) * \frac{0.237}{2} - (F4 + F1) * \frac{0.237}{2}$$



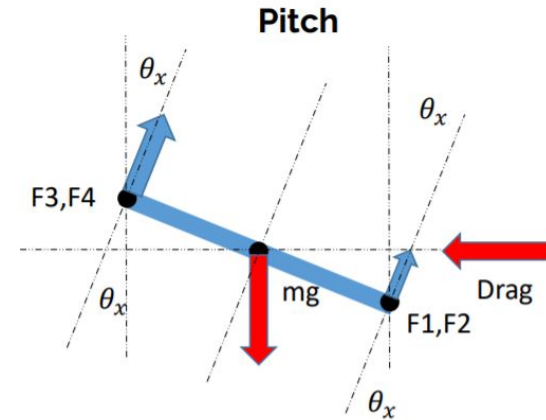
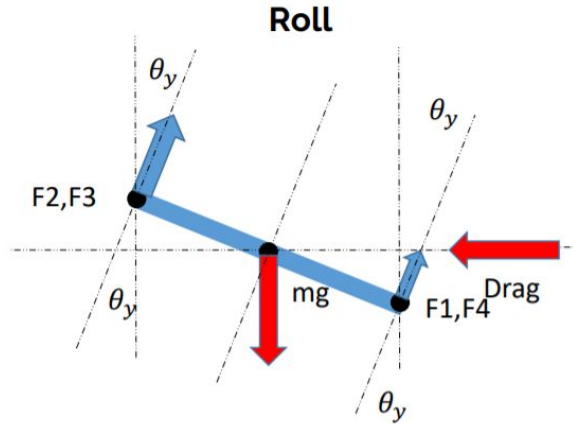
Drag and Disturbances

$$Drag = \frac{1}{2} * \rho * V_{wind}^2 * A * C_d$$

A = 0.0197m², the drag Coefficient C_d of a cube:
1.00 as well as the air density at sea level: $\rho =$
1.225 kg. m⁻³



Roll and Pitch Thrust Vectors



$$F_{prop_x} = \sin(\theta_y) * \cos(\theta_x) * (F1 + F2 + F3 + F4)$$

$$F_{prop_y} = \sin(\theta_x) * \cos(\theta_y) * (F1 + F2 + F3 + F4)$$

$$F_{prop_z} = (F1 + F2 + F3 + F4) * \cos(\theta_x) * \cos(\theta_y)$$

Yaw Thrust Vector

Yaw orients the drone in a certain direction depending on the angle θ_z .

$$\theta_{XY} = -\text{atan2}\left(\frac{F_{prop_x}}{F_{prop_y}}\right)$$
$$XY_{2D} = \sqrt{F_{prop_x}^2 + F_{prop_y}^2}$$

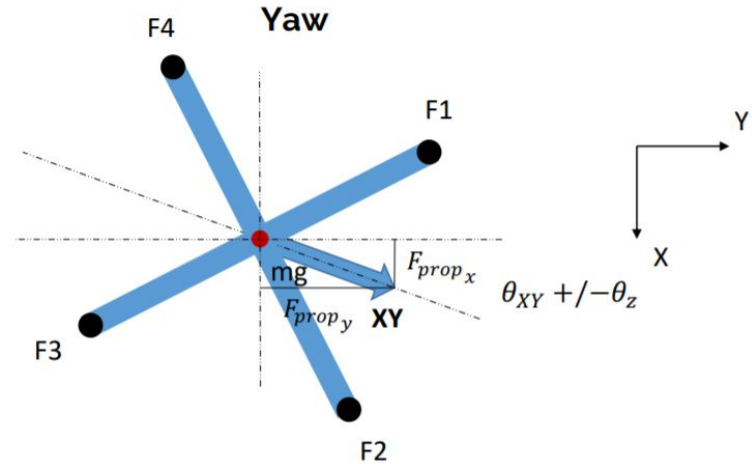
$$F_{prop_x} = XY_{2D} * \sin(\theta_{XY} + /-\theta_z)$$

$$F_{prop_y} = XY_{2D} * \cos(\theta_{XY} + /-\theta_z)$$

$$F_x = F_{prop_x} + /-\text{Drag}_x$$

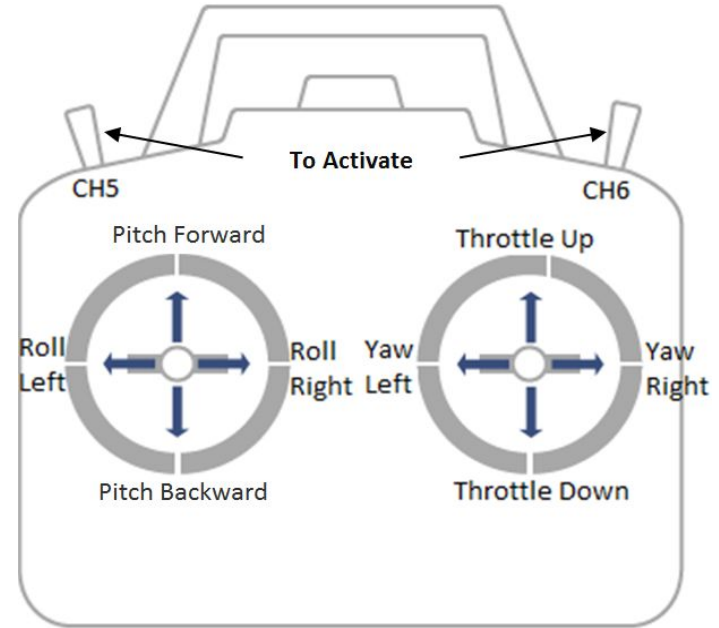
$$F_y = F_{prop_y} + /-\text{Drag}_y$$

$$F_z = F_{prop_z} - mg + /-\text{Drag}_z$$



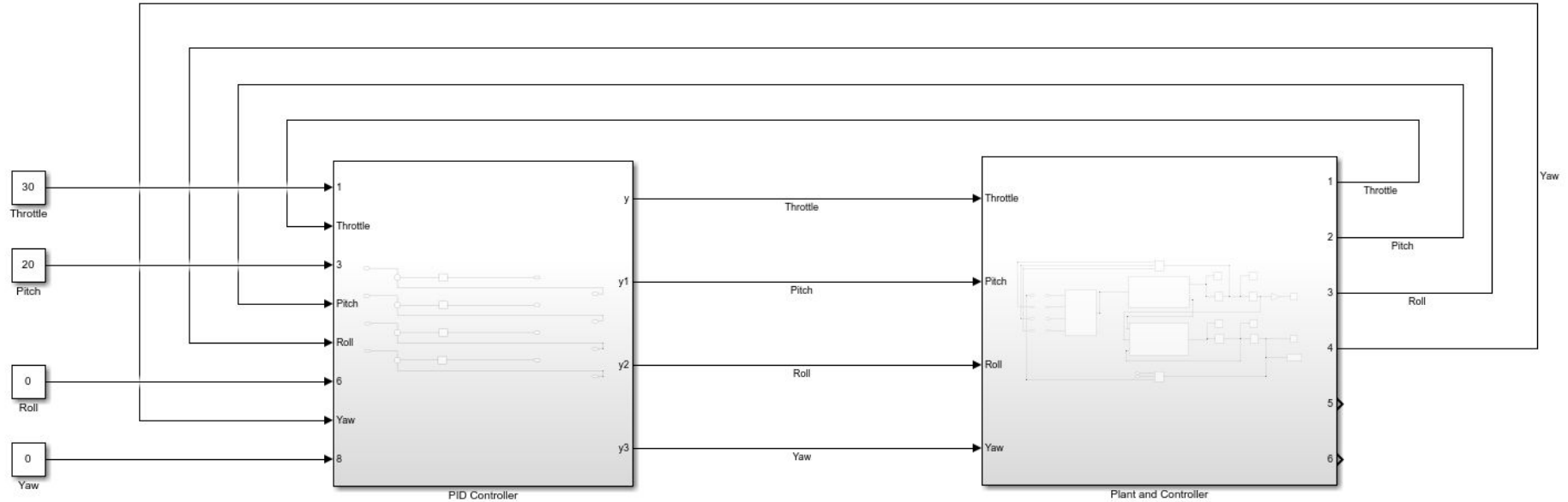
Model Introduction

- The drone will take Thrust, Pitch, Roll and Yaw as inputs from the environment as shown in figure.
- It is a good design practice since whenever someone wants to use drone, then he or she can control the drone more easily.
- The Thrust, Roll, Pitch and Yaw can be commanded independently.

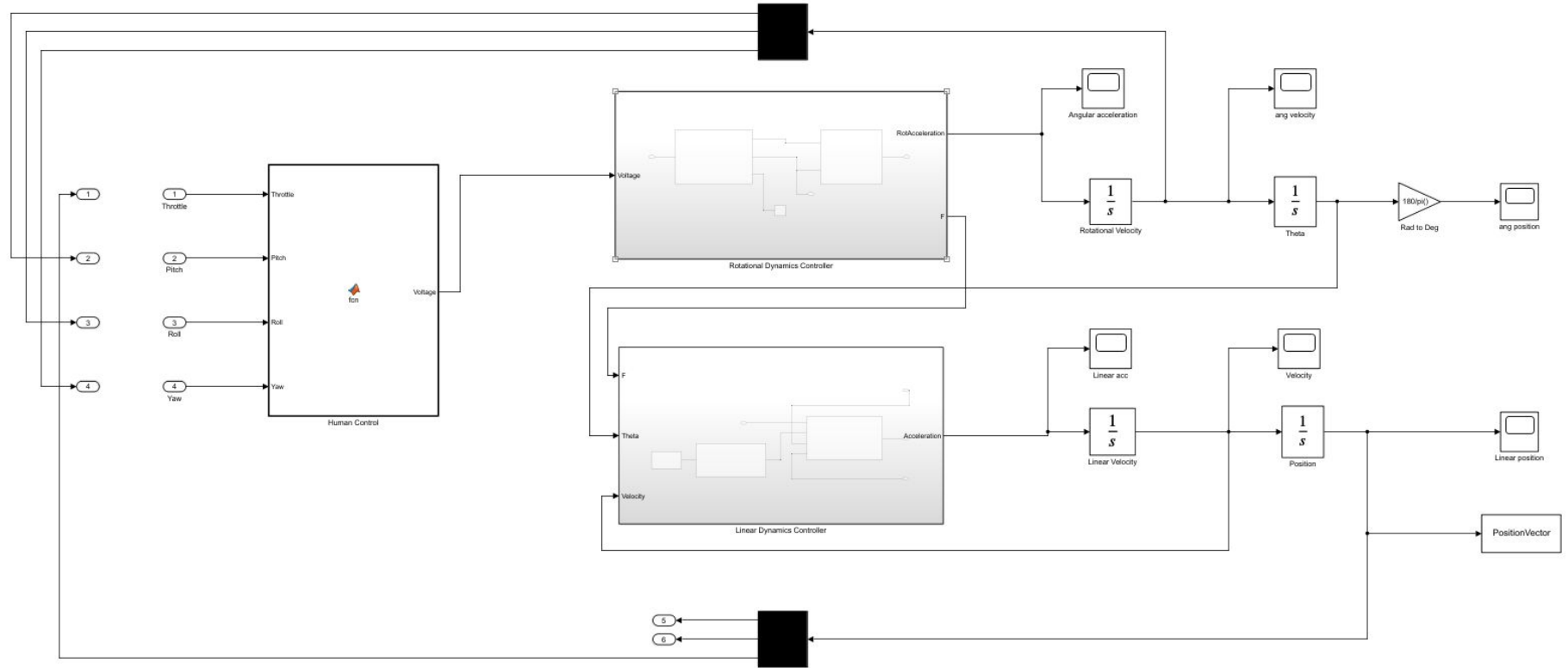


Remote Controller

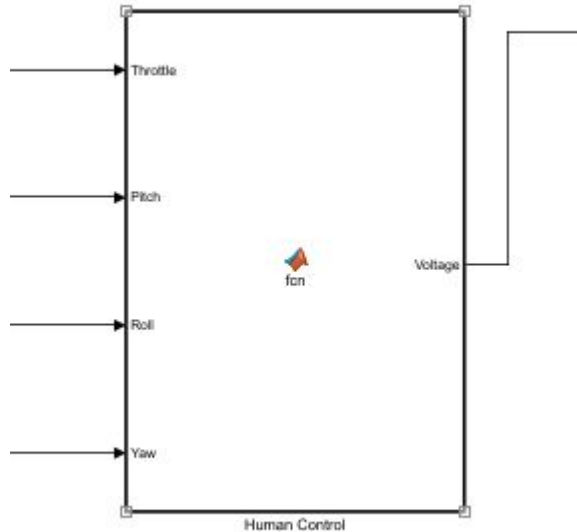
Overall Matlab/Simulink Model Design



Plant and Controller Subsystem

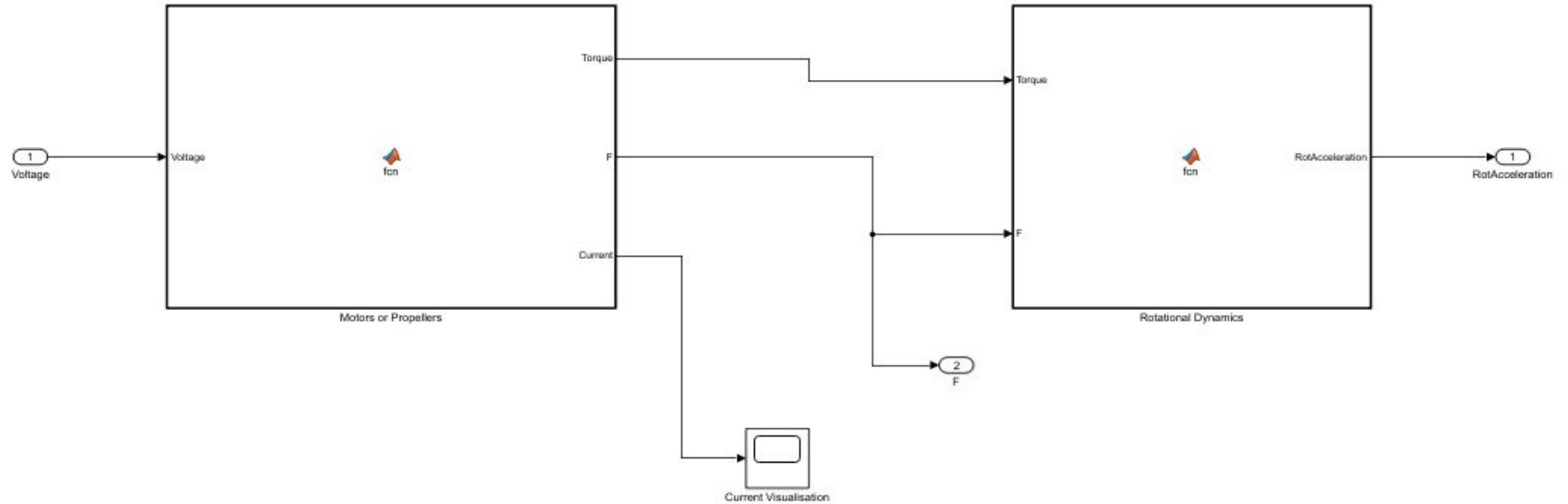


Limiting the Drone Power

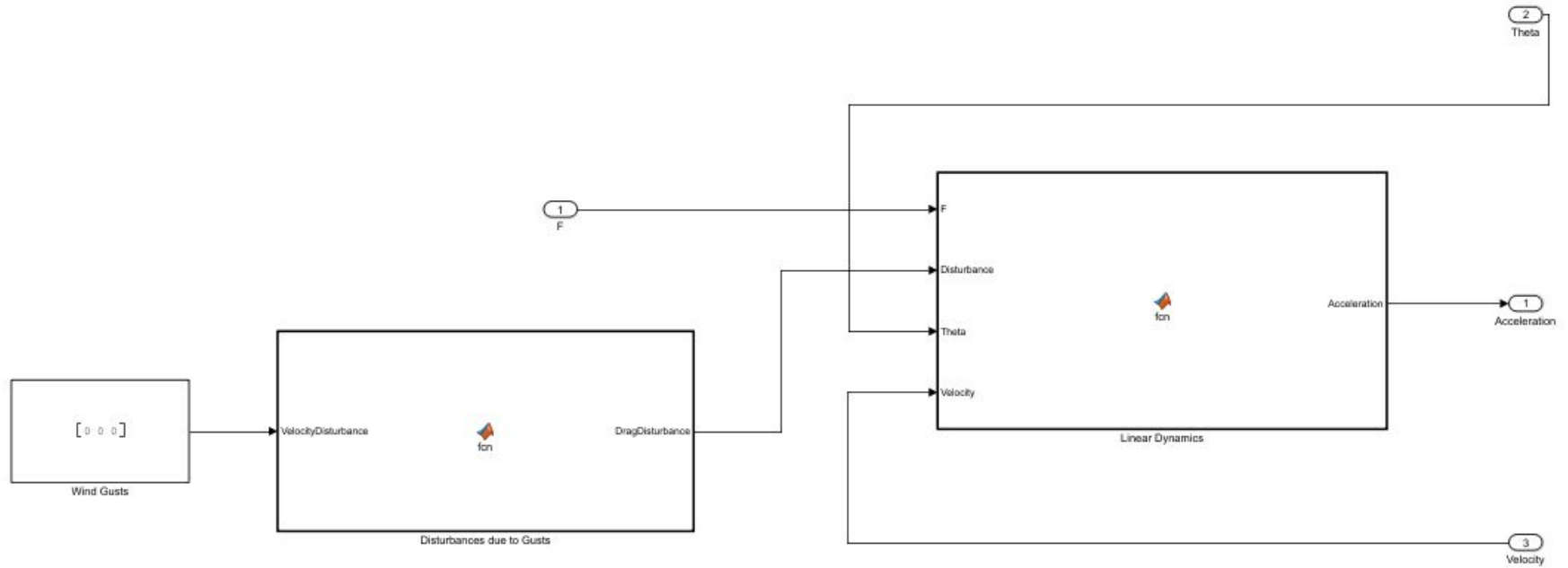


```
Plant and Controller/Human Control  X  +
1  function Voltage = fcn(Throttle, Pitch, Roll, Yaw)
2  Correct = 0;
3  if (Pitch/2 + Roll/2 + Yaw/2) > (100 - Throttle)
4      Correct = ((Pitch/2 + Roll/2 + Yaw/2) - (100 - Throttle));
5  end
6
7  if (Pitch/2 + Roll/2 + Yaw/2) > (Throttle)
8      if Correct < ((Pitch/2 + Roll/2 + Yaw/2) - (Throttle))
9          Correct = ((Pitch/2 + Roll/2 + Yaw/2) - (Throttle));
10     end
11 end
12
13 if Correct ~= 0
14     Pitch = Pitch - Correct/3*2;
15     Roll = Roll - Correct/3*2;
16     Yaw = Yaw - Correct/3*2;
17 end
18
19 Voltage = [(Throttle - Pitch/2 - Roll/2 - Yaw/2)*11.4/100
20            (Throttle - Pitch/2 + Roll/2 + Yaw/2)*11.4/100
21            (Throttle + Pitch/2 + Roll/2 - Yaw/2)*11.4/100
22            (Throttle + Pitch/2 - Roll/2 + Yaw/2)*11.4/100
23            ];
24
```

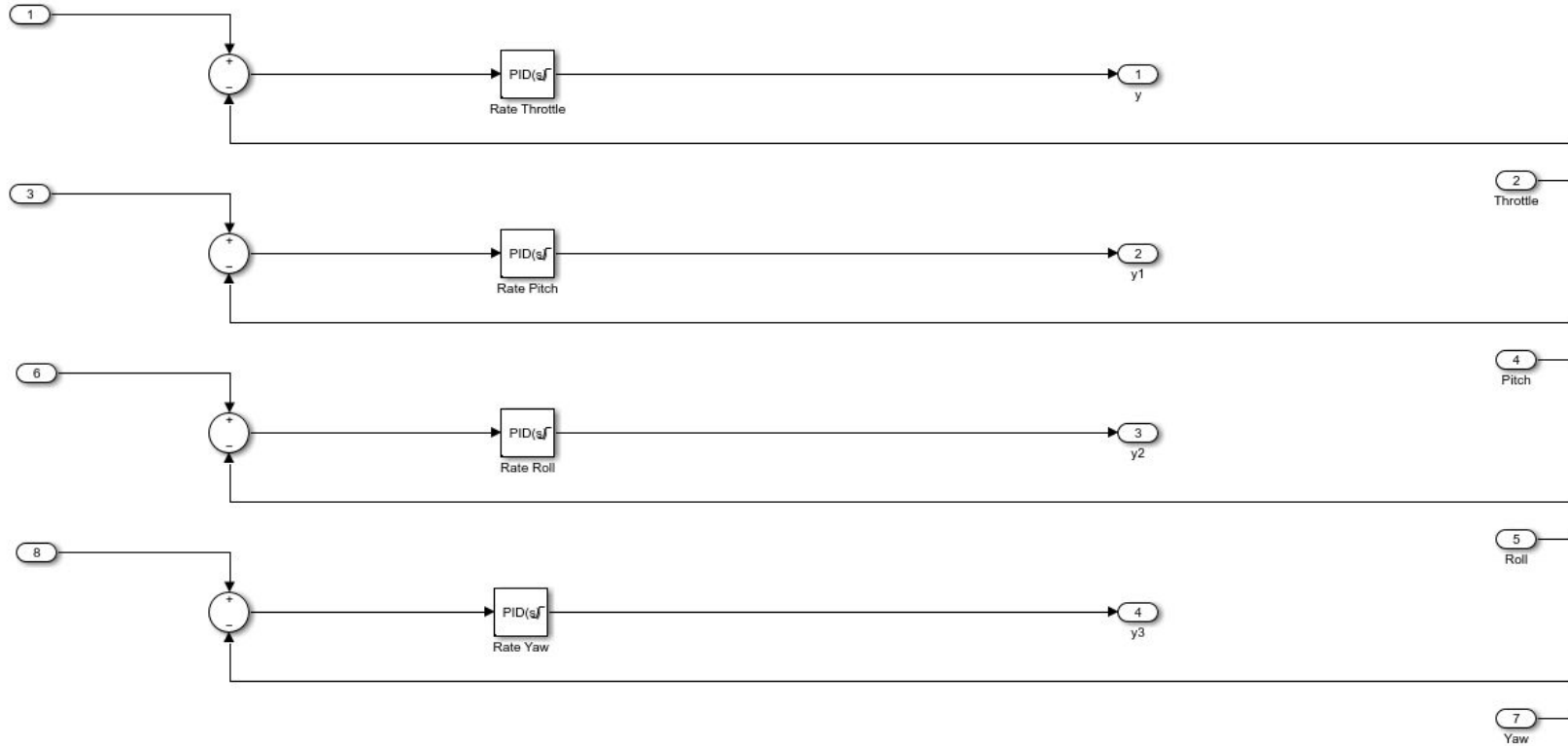
Rotational Dynamics subsystem



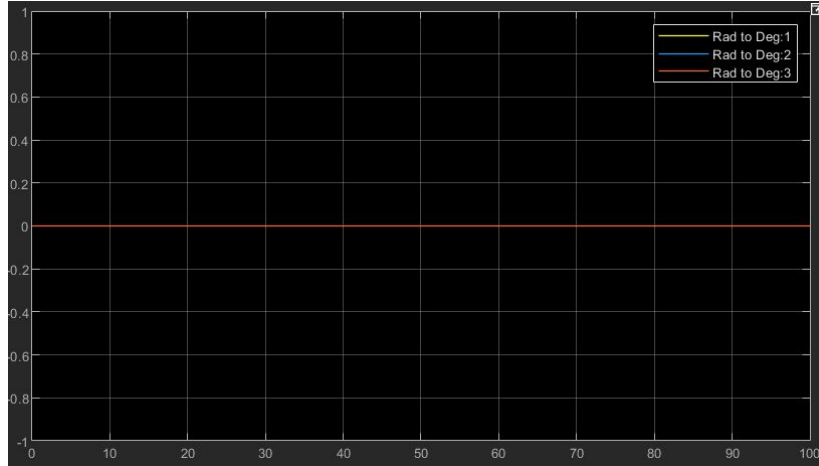
Linear Dynamics Subsystem



PID Controller Design

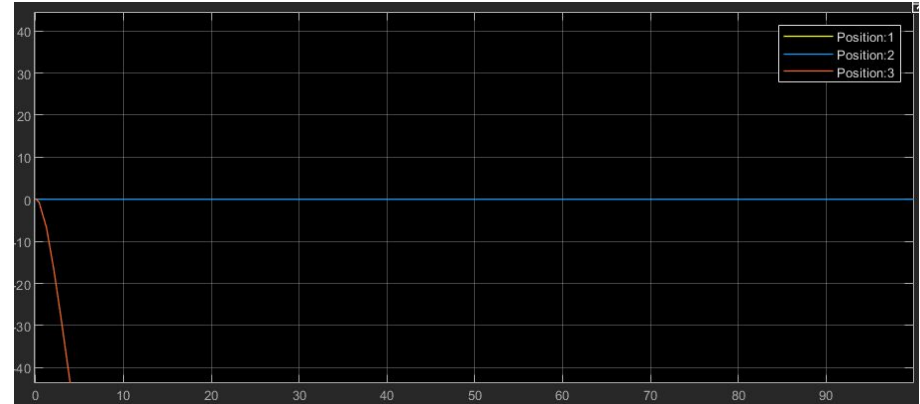


Drone at hover state

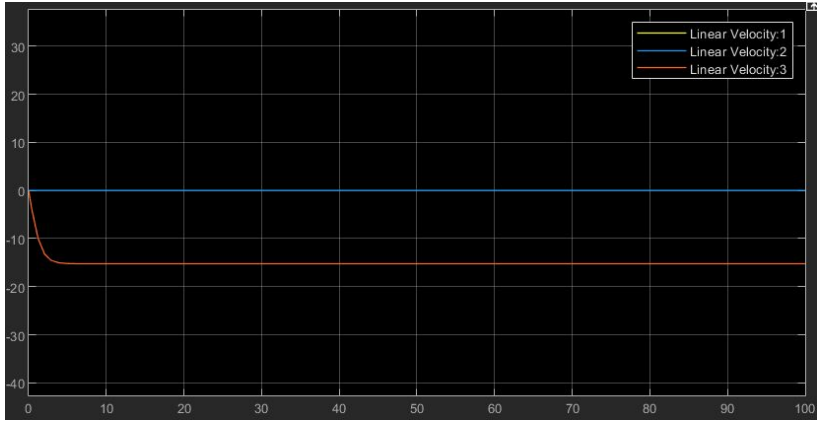


No angular velocity in x, y or z direction

Drone will fall down

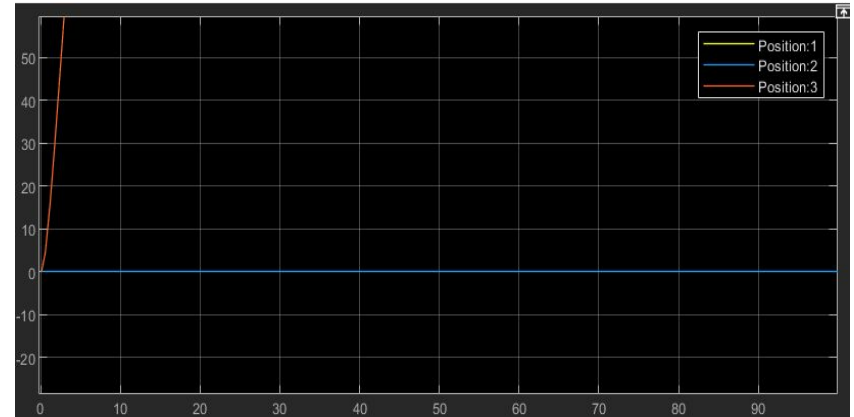


Drone at hover state

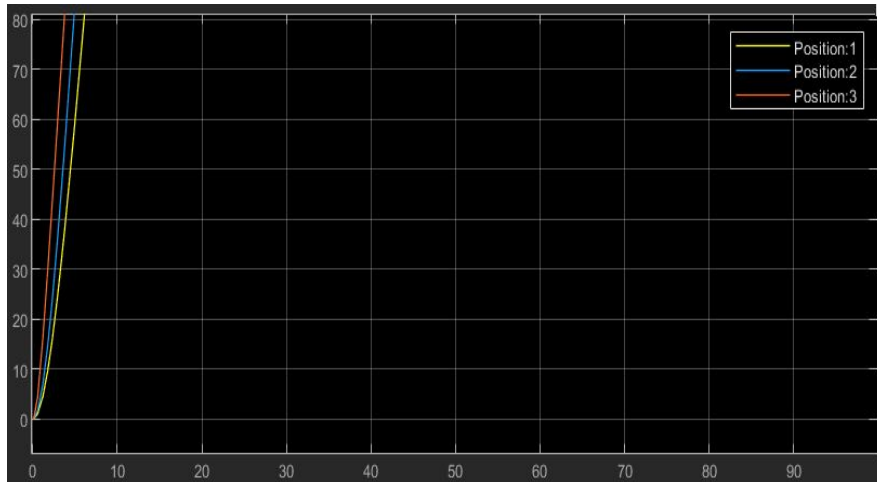


On adding **wind disturbance** in +z direction, drone moves upwards besides the force of gravity (without using PID controller)

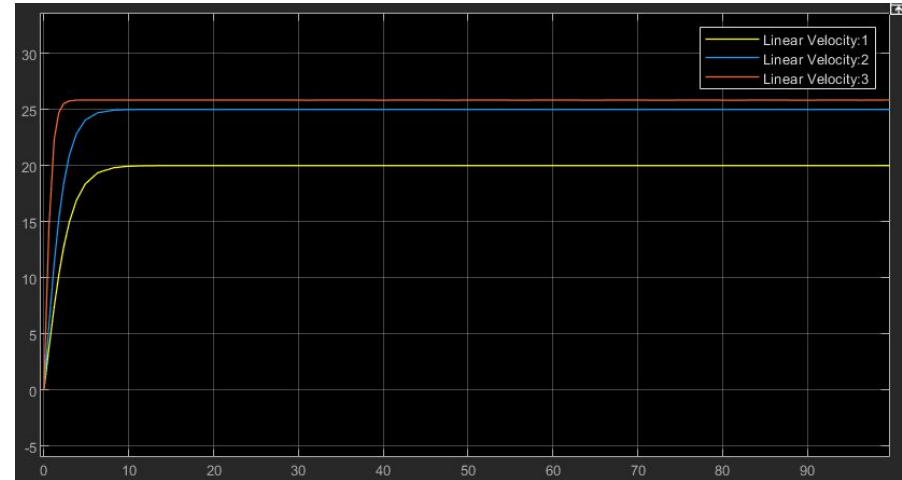
Due to air drag, the drone will eventually reach terminal velocity after sometime.



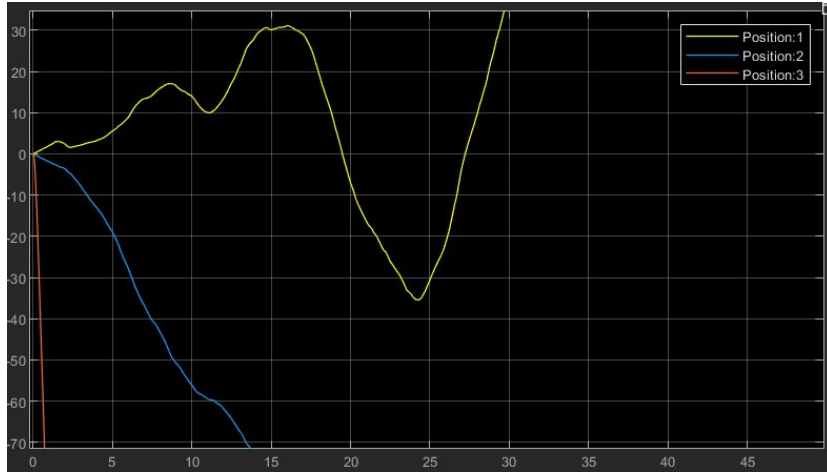
Drone at hover state



Adding wind gusts in 20m/s in +x, 25 m/s in +y and 30 m/s in +z direction leads the drone to move in +x, +y, +z.



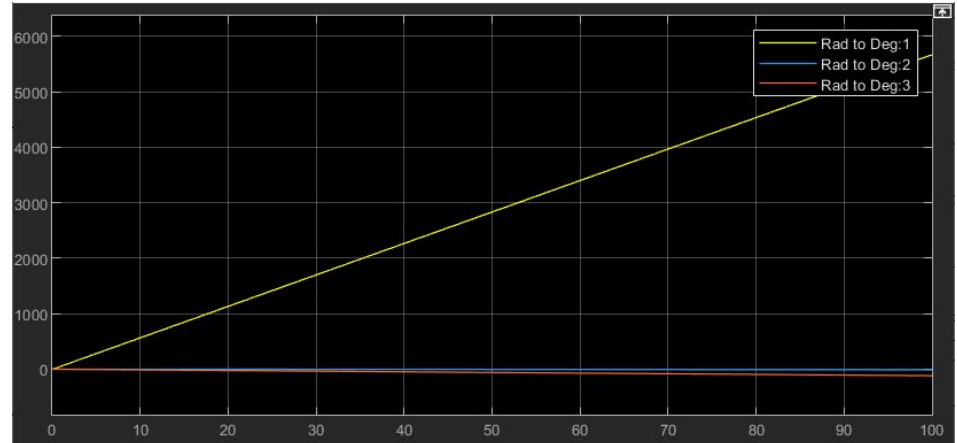
Pitch with Throttle



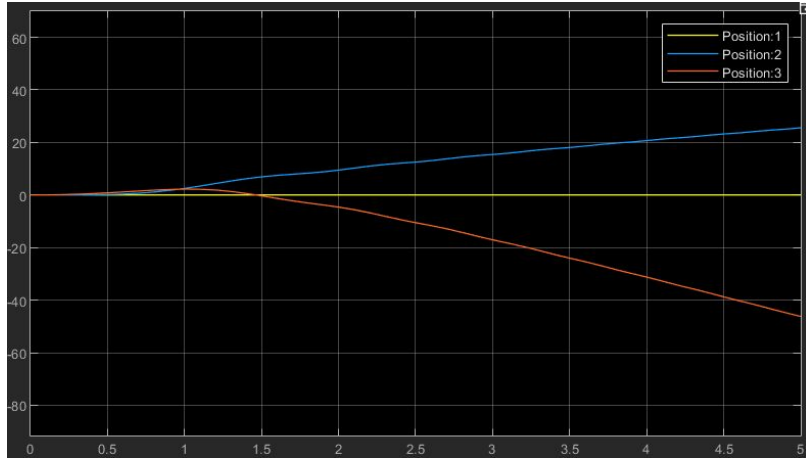
Throttle: 50, Pitch: 1
Roll: 0, Yaw: 0

Rotation about x axis

At too much throttle, 200 here, Drone is falling because maximum throttle is not possible to be achieved.



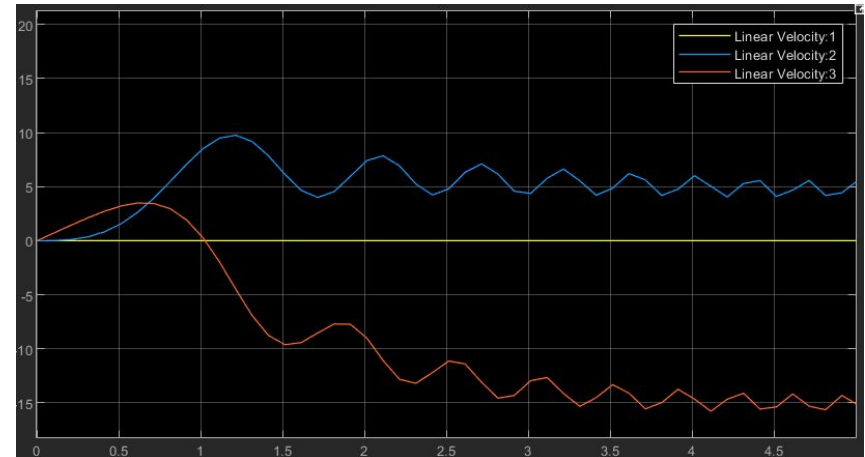
Pitch with Throttle



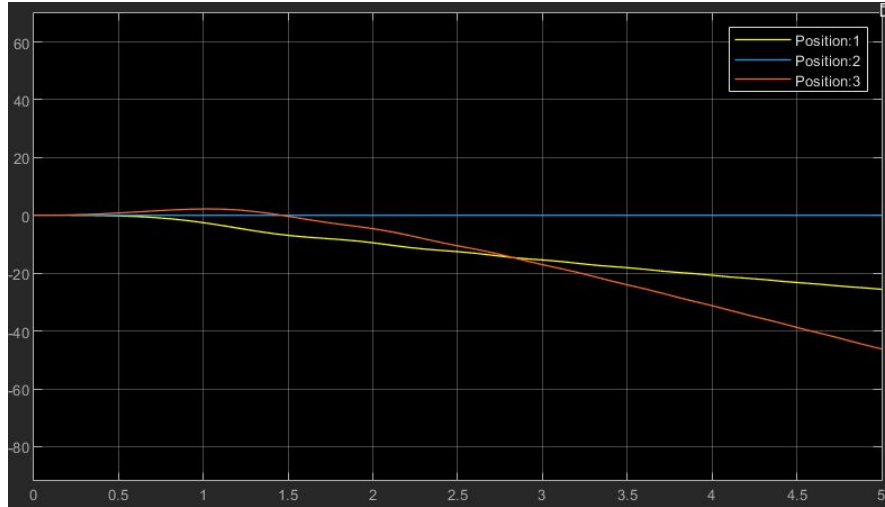
After 180 degrees rotation about x axis at 0.75 seconds, drone will decrease it's velocity in z axis, then after 360 degrees at around 1.5 seconds, it will increase velocity in z and so on. It rotates about 8 rotations about x axis till 5 seconds.

Throttle: 50, Pitch: 1
Roll: 0, Yaw: 0

Drone rotates about x axis, so it will not move in x direction. After 180 degrees rotation about x axis, drone will move in -z direction and +x direction.



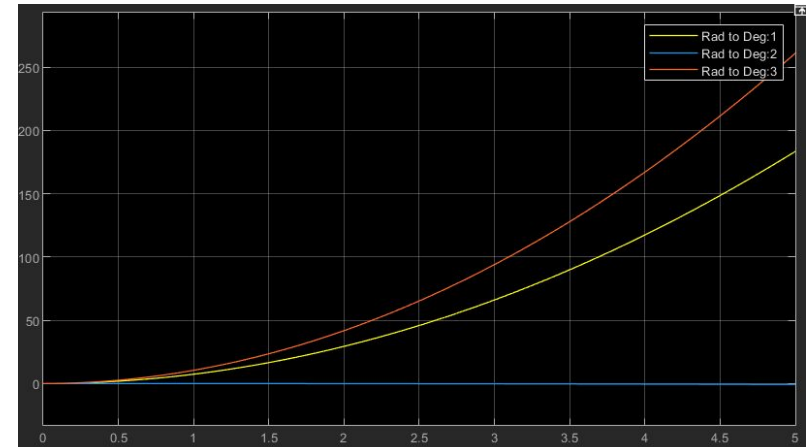
Pitch with Throttle and Yaw



Throttle: 50, Pitch: 0.03
Roll: 0, Yaw: 1

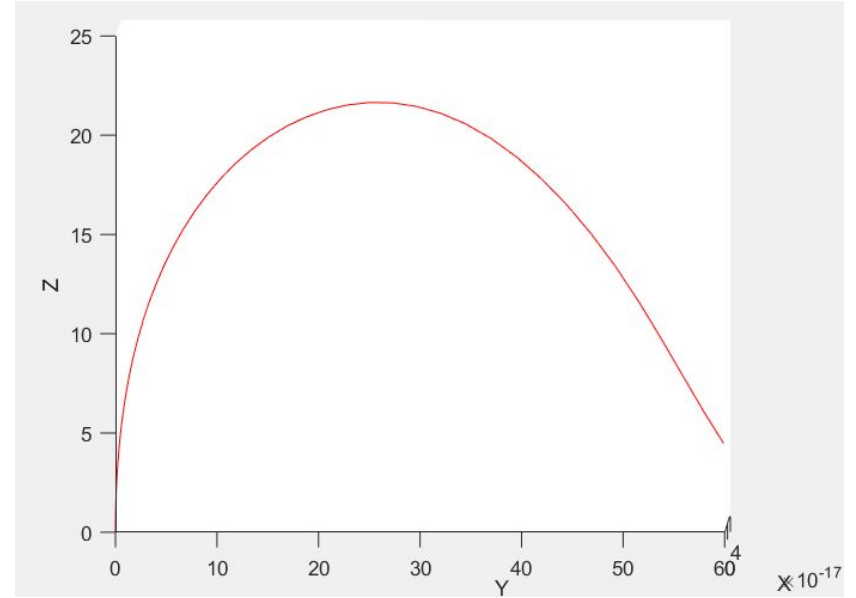
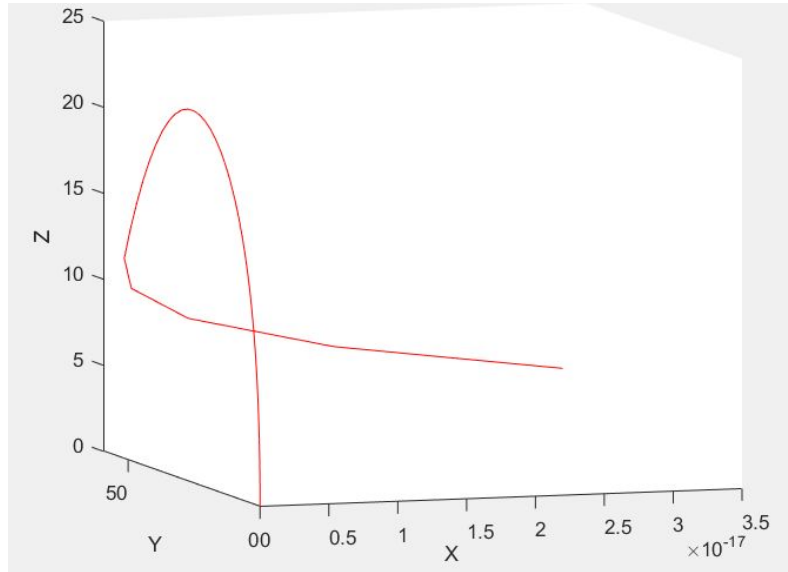
Throttle: 50, Pitch: 0
Roll: 0.5, Yaw: 0

When rotating about y axis, it will first move in +z direction then after each 180 degrees interval, direction is reversed. Similarly for x axis also.



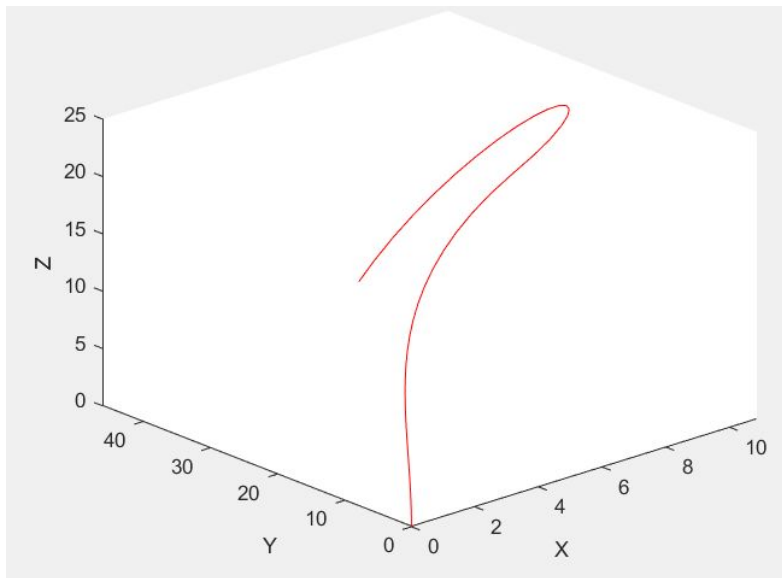
3D Visualization

Throttle: 50, Pitch: 0.035
Roll: 0, Yaw: 0



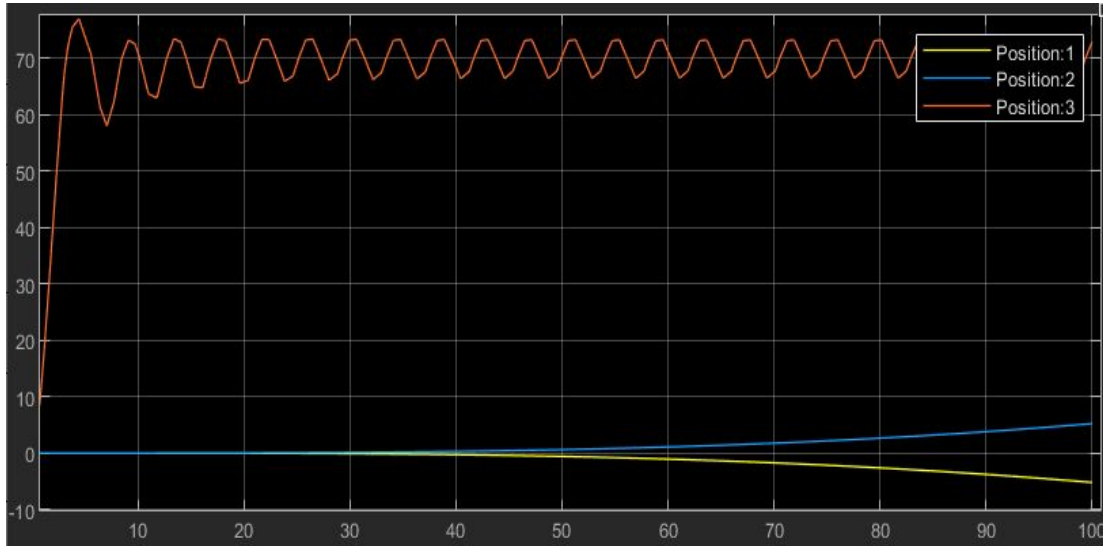
```
>> plot3(PositionVector(:,1), PositionVector(:,2), PositionVector(:,3), '-r')
```

3D Visualization



Throttle: 50, Pitch: 0.035
Roll: 0, Yaw: 0.5

Altitude Control of Drone



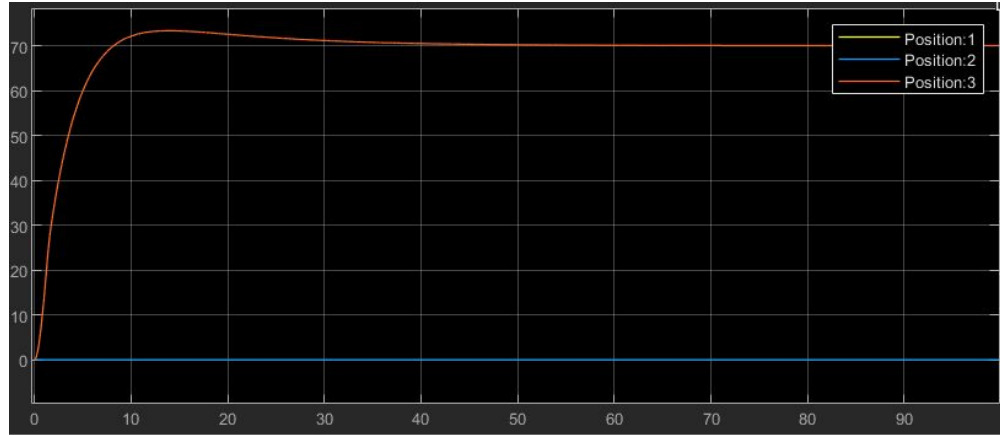
Using PID Controller
without tuning:

$T_p = 5$ (proportional gain)

$T_i = 1$ (integral gain)

$T_d = 0$ (derivative gain)

Altitude Control of Drone



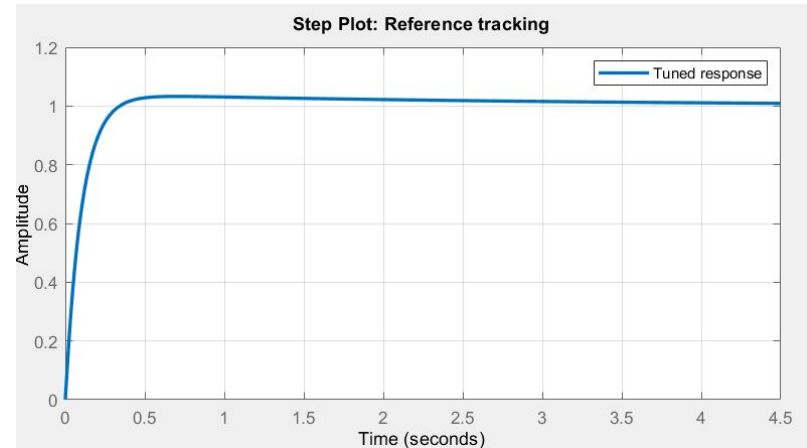
Controller Parameters: $P = 13.2$, $I = 0.8918$, $D = 34.38$, $N = 1143$

To increase speed of response, increase proportional gain.

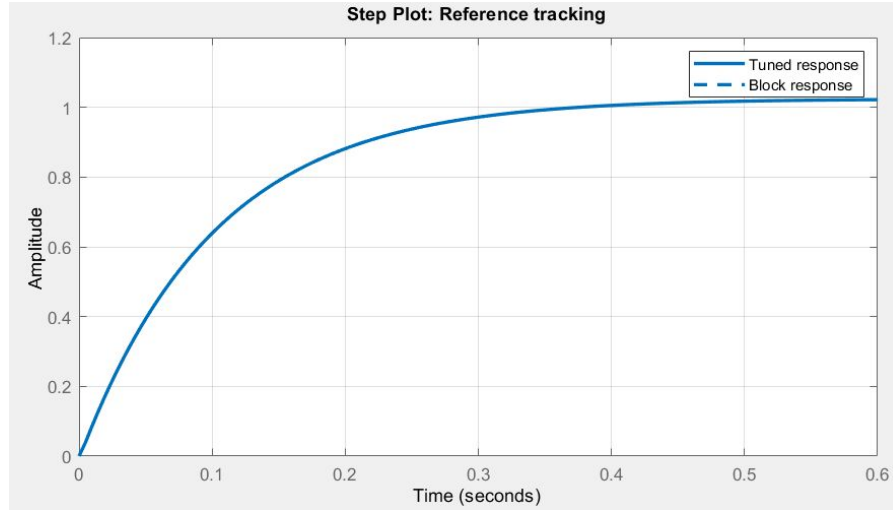
Tuning with PID controller

Throttle = 70.

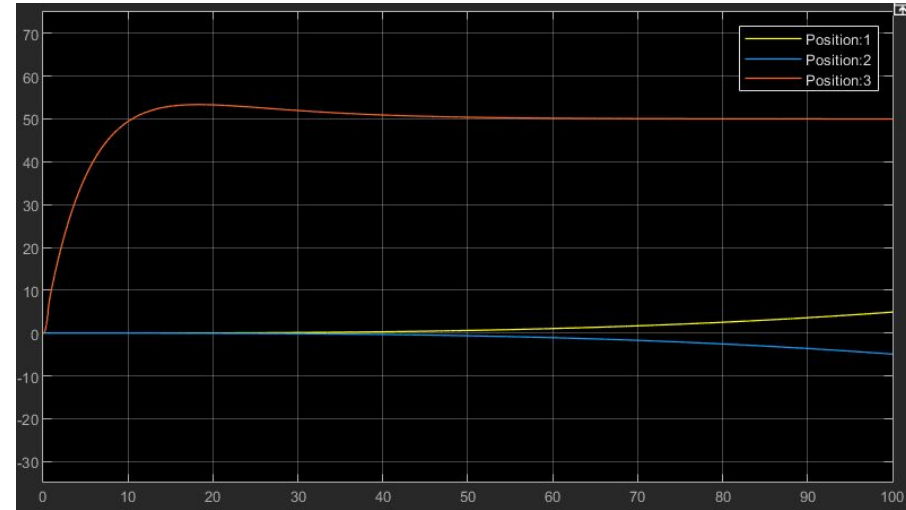
When Throttle is controlled by PID controller, it acts as altitude controller.



Altitude Control with PID Controller

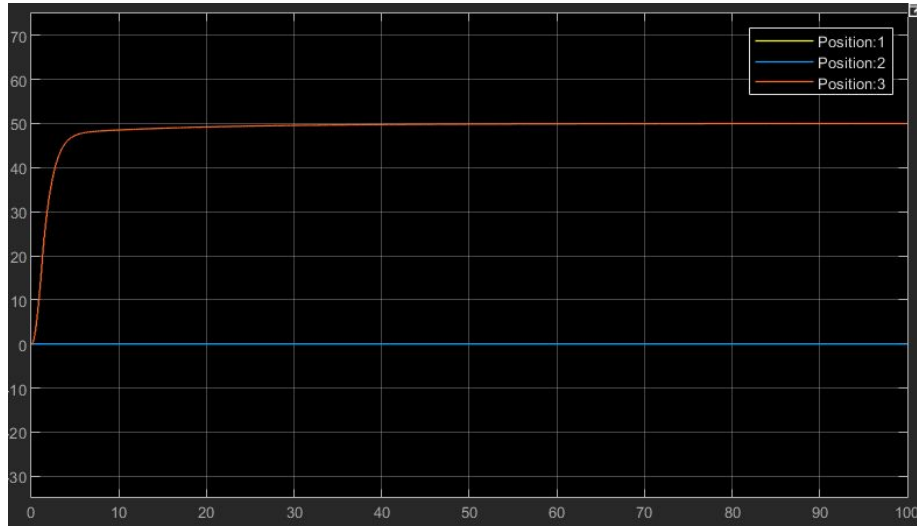


Tuning PID controller while trying not to overshoot.



On increasing derivative gain, system becomes overdamped, it takes more time to come to desired location.

Altitude Control with PID Controller



Proportional (P):

Integral (I):

Derivative (D):

Fig.1: Flat response (do not overshoot at any instance) can be obtained by decreasing the derivative gain after tuning the PID controller.

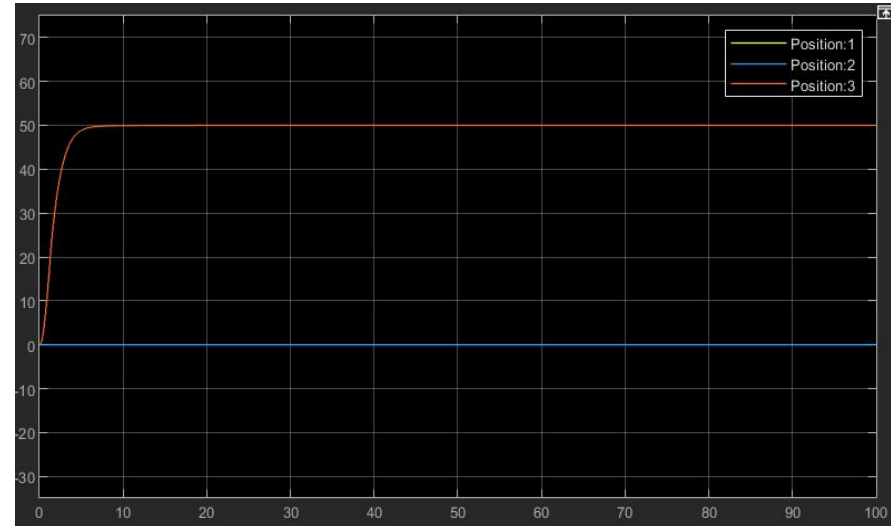
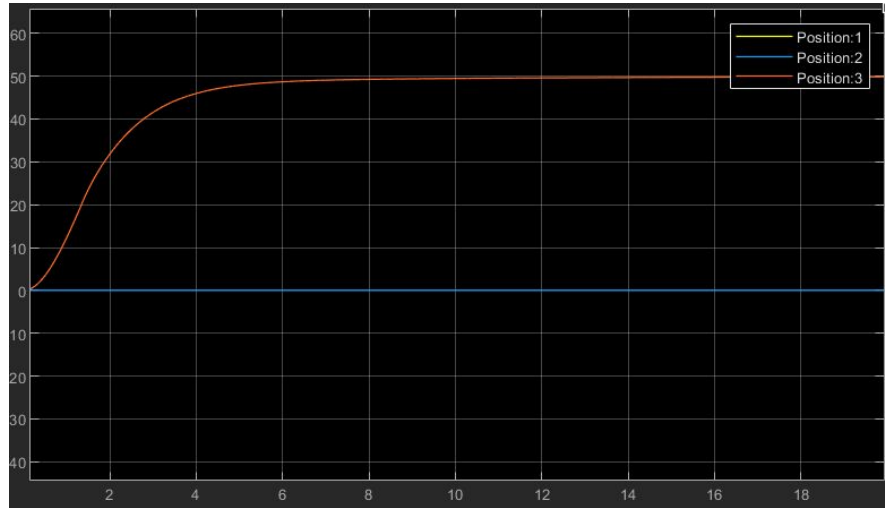
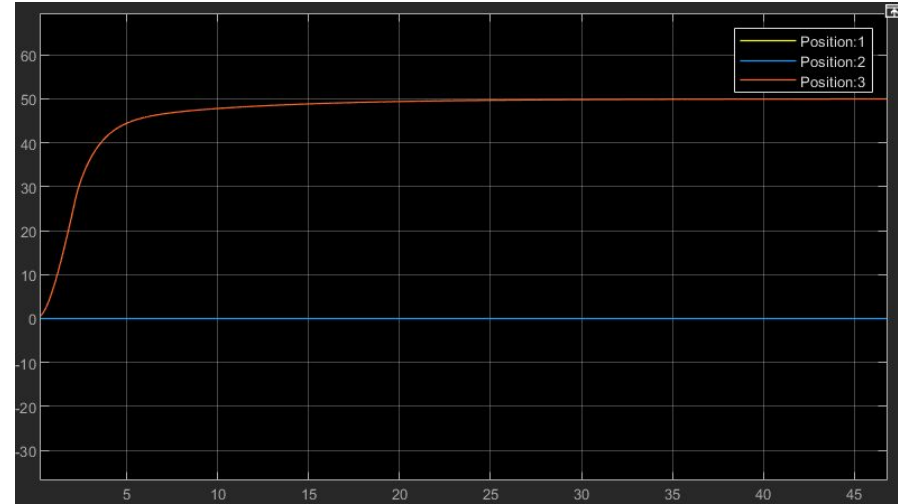


Fig.2: Faster and Flat response can be obtained by increasing integral gain and decreasing derivation gain after tuning the PID controller.

Control with Altitude PID Controller and Air Gusts



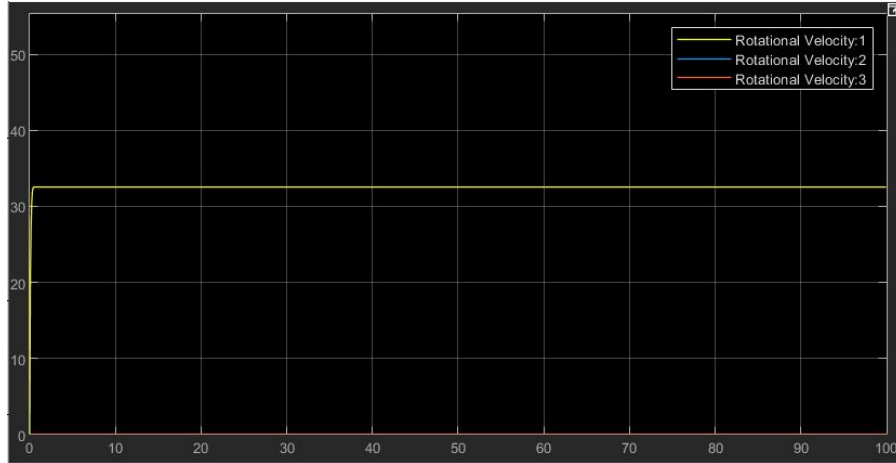
Wind disturbance in +z: -10 m/s
Taking around 8 seconds to reach



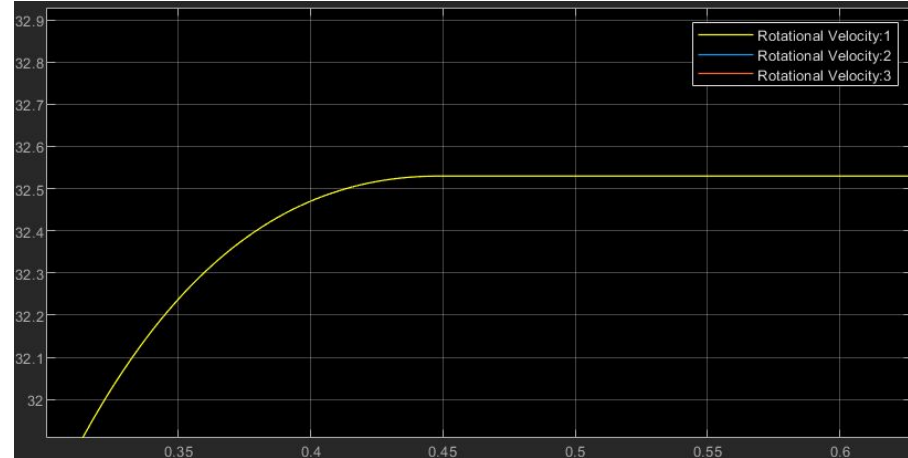
Wind disturbance in +z: -20 m/s
Taking about 15 seconds to reach

Pitch PID Controller

- For rotational rate control, tune Pitch, Roll and Yaw.

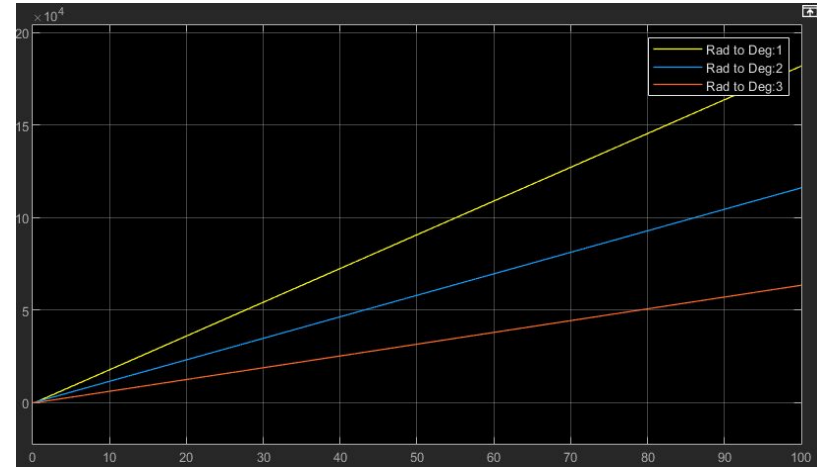
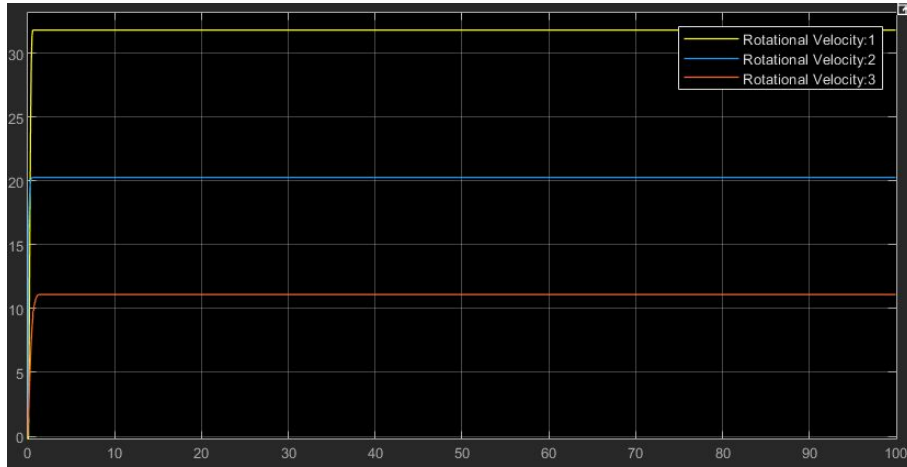


Throttle: 50, Pitch: 32.5, Roll: 0, Yaw: 0



32.5 m/s is obtained after 0.45 seconds of movement

Pitch, Roll and Yaw PID Controller



Throttle: 50, Pitch: 30, Roll: 20, Yaw: 10

Having steady state errors in Yaw and Pitch directions but they can be tuned.

Formal Verification using KeYmaera X

CS591

```
1 ArchiveEntry "CS591"  
2 Description "UAV_Drone".  
3
```

```
4 /*  
5   Air Drags and Air Disturbances are neglected.  
6   Motor or Propellers limits are also neglected.  
7 */  
8 /*
```

```
9   a = Sin(thetax)  
10  b = Cos(thetax)  
11  c = Sin(thetay)  
12  d = Cos(thetay)
```

```
13  
14  a' = b  
15  b' = -a  
16  c' = d  
17  d' = -c
```

```
18  
19  e = Sin(thetaz)  
20  f = Cos(thetaz)  
21  g = Sin(thetaXY)  
22  h = Cos(thetaXY)
```

```
23  
24  e' = f  
25  f' = -e  
26  g' = h  
27  h' = -g  
28 */
```

```
29  
30 /* Represent all the constant terms used */  
31
```

Definitions

```
32 Real Ax, Ay, Az;  
33 Real m, g, Ix, Iy, Iz;  
34 Real rho, Cd;
```

End.

```
37  
38 /* Represent all the state variables */  
39
```

ProgramVariables

```
40 Real throttle, pitch, roll, yaw;  
41 Real voltage1, voltage2, voltage3, voltage4, RPM1, RPM2, RPM3, RPM4;  
42 Real Torque1, Torque2, Torque3, Torque4;  
43 Real F1, F2, F3, F4, Momentx, Momenty, Momentz, Forcex, Forcey, Forcez;  
44 Real Fprop1, Fprop2, Fprop3;  
45 Real x, y, vx, vy, vz, Accx, Accy, Accz;  
46 Real thetax, thetay, thetaz, RotAccx, RotAccy, RotAccz;  
47 Real ThetaXY, XY2D;
```

End.

```
50
```

Formal Verification using KeYmaera X

```
51 Problem
52 /* All the input values are given here */
53 throttle = 30 & pitch = 0 & roll = 0 & yaw = 0 & Ix = 0.003 & Iy = 0.003 & Iz = 0.007 & m = 0.734 & g = 9.81 &
54 Ax = 0.0197 & Ay = 0.0197 & Az = 0.0512 & rho = 1.225 & Cd = 1
55 ->
56 [{
57   /* Represent the continuous part of dynamics */
58   {
59     x' = vx, y' = vy, throttle' = vz, vx' = Accx, vy' = Accy, vz' = Accz,
60     thetax' = pitch, thetay' = roll, thetaz' = yaw, pitch' = RotAccx, roll' = RotAccy, yaw' = RotAccy,
61     a' = b, b' = -1*a, c' = d, d' = -1*c,
62     e' = f, f' = -1*e, g' = h, h' = -1*g,
63     ThetaXY' = -1 / (1 + (Fprop1/Fprop2)^2)
64   }
65   /* Represent the discrete part of dynamics */
66   {
67     /* Motors or Propellers */
68     voltage1 := (throttle - pitch/2 - roll/2 - yaw/2)*11.4/100;
69     voltage2 := (throttle - pitch/2 + roll/2 + yaw/2)*11.4/100;
70     voltage3 := (throttle + pitch/2 + roll/2 - yaw/2)*11.4/100;
71     voltage4 := (throttle + pitch/2 - roll/2 + yaw/2)*11.4/100;
72
73     RPM1 := -2.6931*(Voltage1^3) + 1400*Voltage1;
74     RPM2 := -2.6931*(Voltage2^3) + 1400*Voltage2;
75     RPM3 := -2.6931*(Voltage3^3) + 1400*Voltage3;
76     RPM4 := -2.6931*(Voltage4^3) + 1400*Voltage4;
77
78     F1 := (2*(10^-15)*(RPM1^3) - 4*(10^-11)*(RPM1^2) + 3*(10^-7)*RPM1 + 0.1013)*1.225*((RPM1/60)^2)*0.0016;
79     F2 := (2*(10^-15)*(RPM2^3) - 4*(10^-11)*(RPM2^2) + 3*(10^-7)*RPM2 + 0.1013)*1.225*((RPM2/60)^2)*0.0016;
80     F3 := (2*(10^-15)*(RPM3^3) - 4*(10^-11)*(RPM3^2) + 3*(10^-7)*RPM3 + 0.1013)*1.225*((RPM3/60)^2)*0.0016;
81     F4 := (2*(10^-15)*(RPM4^3) - 4*(10^-11)*(RPM4^2) + 3*(10^-7)*RPM4 + 0.1013)*1.225*((RPM4/60)^2)*0.0016;
82
83     Torque1 := 4*(10^-14)*(RPM1^3) + 8*(10^-12)*(RPM1^2) + 3*(10^-6)*RPM1;
84     Torque2 := 4*(10^-14)*(RPM2^3) + 8*(10^-12)*(RPM2^2) + 3*(10^-6)*RPM2;
85     Torque3 := 4*(10^-14)*(RPM3^3) + 8*(10^-12)*(RPM3^2) + 3*(10^-6)*RPM3;
86     Torque4 := 4*(10^-14)*(RPM4^3) + 8*(10^-12)*(RPM4^2) + 3*(10^-6)*RPM4;
87   }
```


Formal Verification using KeYmaera X

```
88      /* Rotational Dynamics */
89      Moment1 := (F3 + F4)*0.237/2 - (F1 + F2)*0.237/2;
90      Moment2 := (F3 + F2)*0.237/2 - (F1 + F4)*0.237/2;
91      Moment3 := Torque4 - Torque1 + Torque2 - Torque3;
92
93      RotAccx := Momentx/Ix;
94      RotAccy := Momenty/Iy;
95      RotAccz := Momentz/Iz;
96
97      /* Linear Dynamics */
98      Fprop1 := c * b * (F1 + F2 + F3 + F4);
99      Fprop2 := a * d * (F1 + F2 + F3 + F4);
100     Fprop3 := b * d * (F1 + F2 + F3 + F4);
101
102     XY2D := sqrt(Fprop1^2 + Fprop2^2);
103
104     ?Fprop3 >= 0; Fprop1 := XY2D * (e*h + f*g); ++
105     ?Fprop3 >= 0; Fprop2 := XY2D * (f*h - e*g); ++
106     ?Fprop3 < 0; Fprop1 := XY2D * (g*f - e*h); ++
107     ?Fprop3 < 0; Fprop2 := XY2D * (e*g + f*h);
108
109     Forcex := F1;
110     Forcey := F2;
111     Forcez := F3 - m * g;
112
113     Accx := Forcex / m;
114     Accy := Forcey / m;
115     Accz := Forcez / m;
116   }
117 }*@invariant(throttle>=25 & throttle<=35)] throttle>=25 & throttle<=35
118 End.
119
120 Tactic "UAV_Drone: automatic"
121   auto
122 End.
123
124 End.
```

References

Modelling and Simulation:

- https://www.youtube.com/watch?v=iS5JFuopQsA&ab_channel=VDEngineering
- <https://in.mathworks.com/videos/drone-simulation-and-control-part-1-setting-up-the-control-problem-1539323440930.html>
- <https://www.apcprop.com/technical-information/performance-data/>
- https://www.apcprop.com/files/PER3_8x4.dat

Formal Verification:

- <https://keymaerax.org/>
- <https://www.cs.cmu.edu/~smitsch/pdf/KeYmaera-tutorial.pdf>
- <https://youtu.be/v546o8TVN1w>

My overall work can be seen here:

- https://github.com/SAMAYV/CS591_Simulation_Verification

Thank you