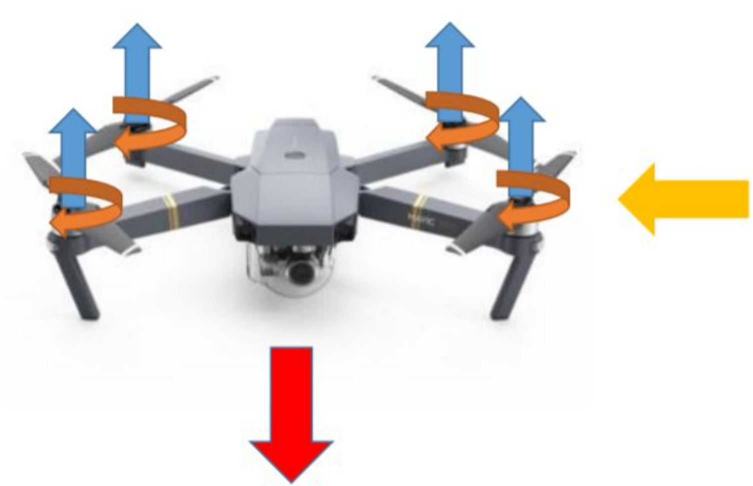
CS591
UAV or Drone Simulation

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

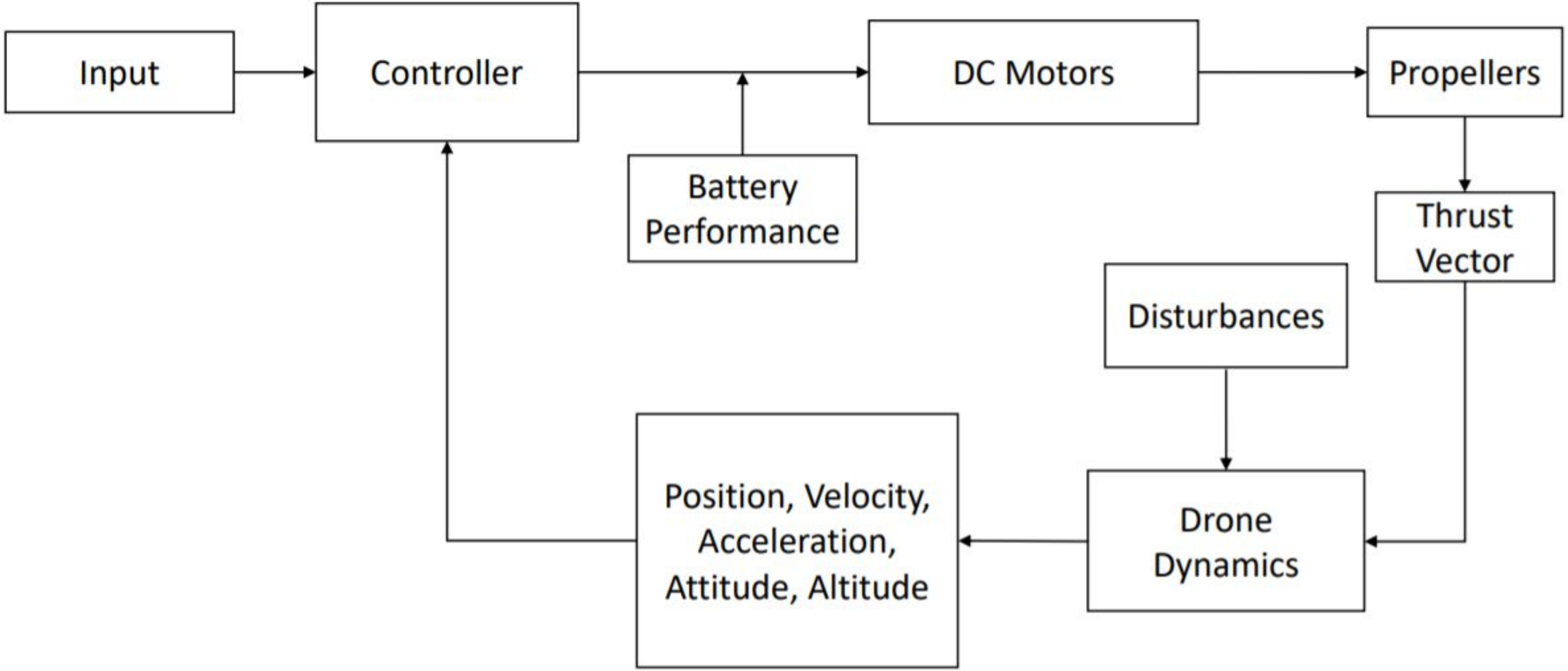
: Gravity

: 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.

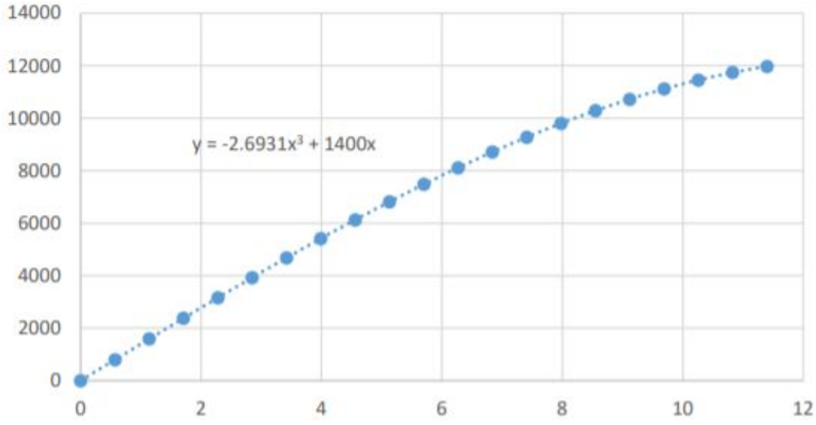
Simplified Model Diagram



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.



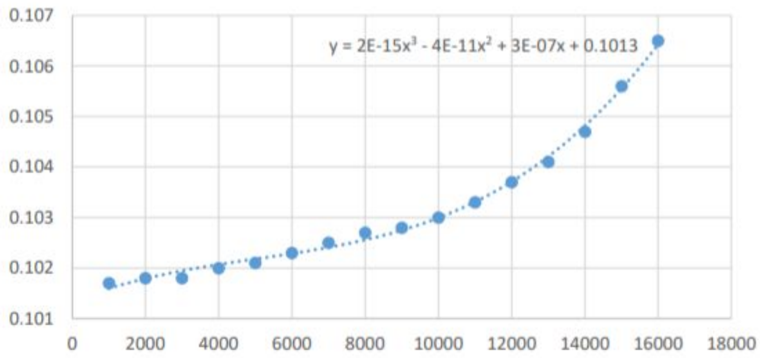
$$RPM = -2.6931 * V^3 + 1400 * V$$

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.



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

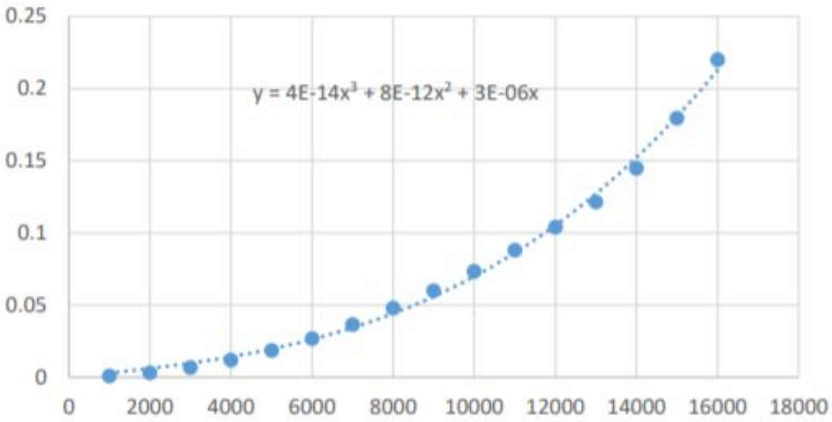
Torque Equation

- 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 KV.

$$Torque = K_T * I$$

$$K_T = \frac{1}{K_V} = \frac{1}{1400} = 0.007$$

$$I = Torque * 1400$$



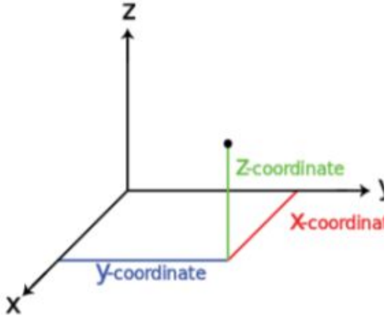
$$Torque = 4 * 10^{-14} * RPM^3 + 8 * 10^{-12} * RPM^2 + 3 * 10^{-6} * RPM$$

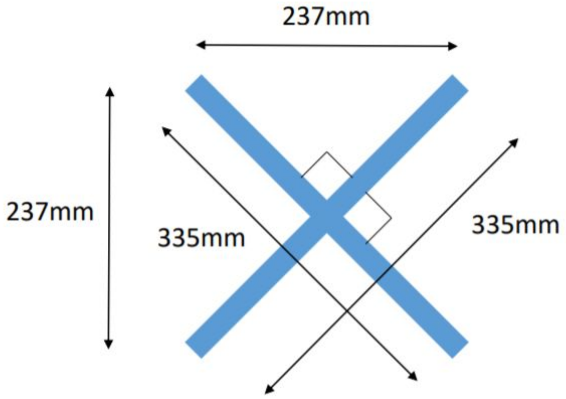
Shape of Drone and Axes

In the x direction: $F_x = ma_x$
In the y direction: $F_y = ma_y$
In the z direction: $F_z = ma_z$

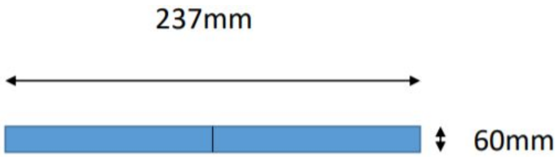
About the x axis (pitch): $M_x = I_x * \ddot{\theta}_x$
About the y axis (roll): $M_y = I_y * \ddot{\theta}_y$
About the z axis (yaw): $M_z = I_z * \ddot{\theta}_z$

M is the external moments or torques in Nm, I is the moments of inertia in kg.m² ar $\ddot{\theta}$ is rotational acceleration in rad/s².





Top View



Side View

Moment of Inertia

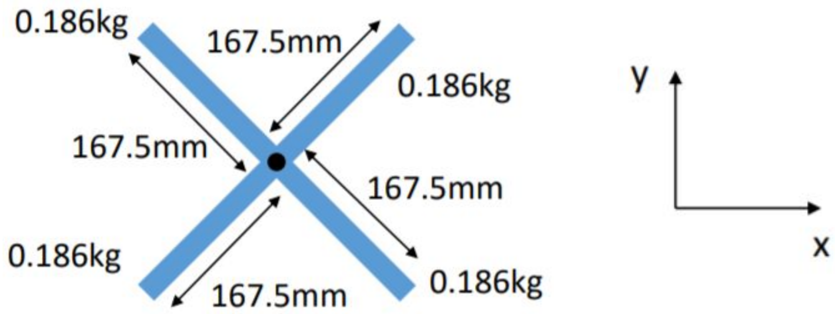
Moment of Inertia of a rod about its end:

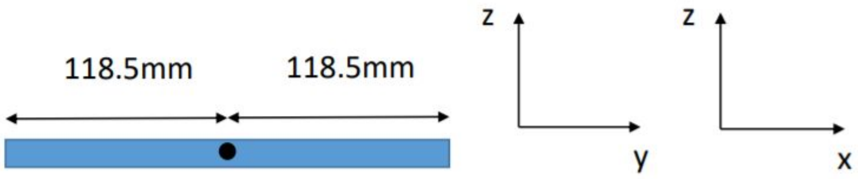
$$I = \frac{1}{3}ML^2$$

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

$$I_z = 0.007\ kg.m^2$$

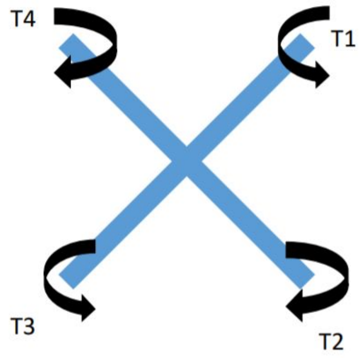
$$I_{x,y} = \frac{1}{3} * \frac{0.743}{4} * \left(\frac{0.237}{2}\right)^2 * 4 = 0.003\ kg.m^2$$





Torques and Propellers Rotation

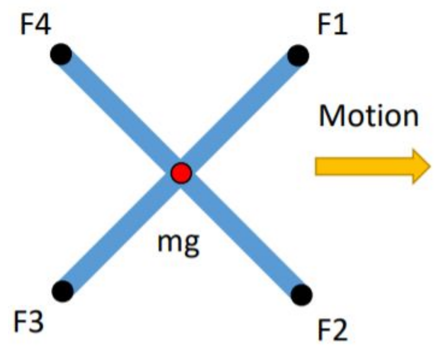
- The torques necessary to rotate each propellers also act on the drone.
- Props 2 and 4 rotate in the same direction and oppositely to 1 and 3.
- Decreasing or increasing the power in each of these couples independently induces yaw.



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

Thrust and Moments

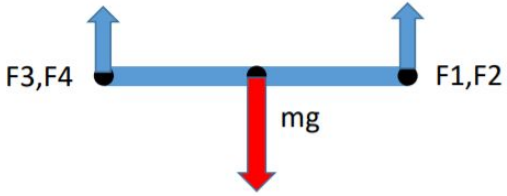
- F1, F2, F3 and F4 each represent the thrust of their respective propeller.
- F1 and F2 as well as F4 and F3 are respectively coupled to induce pitch.
- F4 and F1 as well as F3 and F2 are respectively coupled to induce roll.

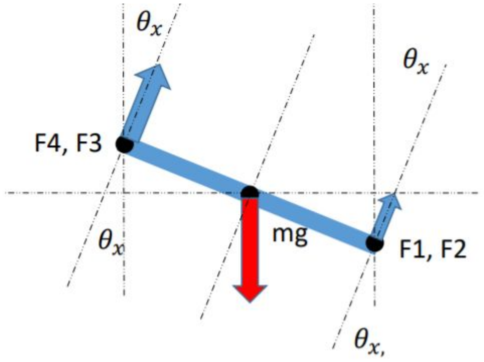


Pitch and Roll Motions

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

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

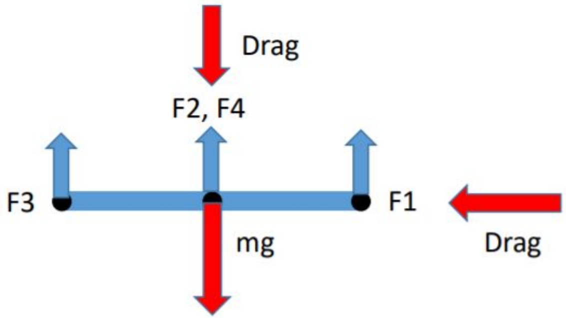




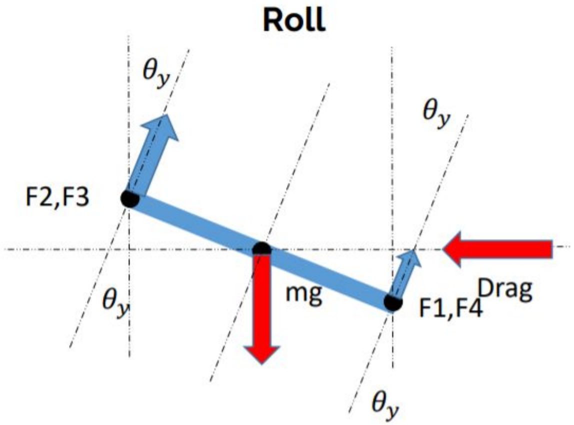
Drag and Disturbances

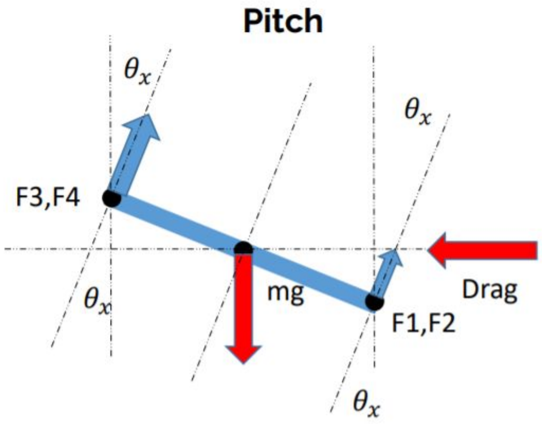
$$Drag = \frac{1}{2} * \rho * V_{wind}^2 * A * C_d$$

A = 0.0197m², the drag Coefficient C_d of a cube: 1.00 as well as the air density at sea level: ρ = 1.225 $kg.m^{-3}$



Roll and Pitch Thrust Vectors





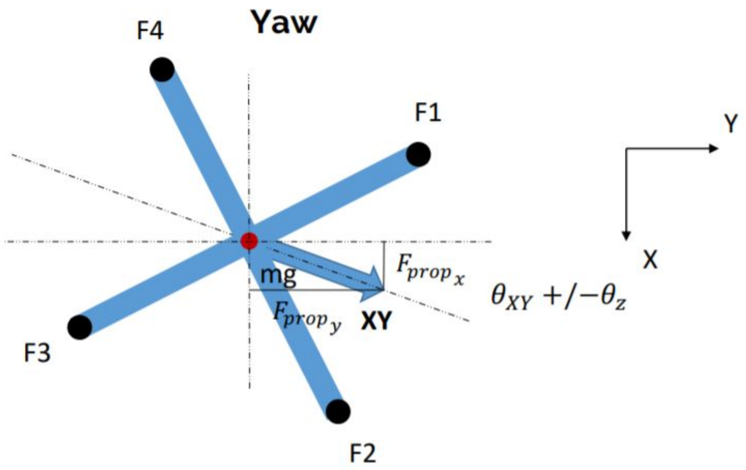
$$F_{prop_x} = \sin(\theta_y) * \cos(\theta_x) * (F1 + F2 + F3 + F4)$$

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

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

Yaw Thrust Vector

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



$$\theta_{XY} = -\text{atan2}(\frac{F_{prop_x}}{F_{prop_y}})$$

$$XY_{2D} = \sqrt{F_{prop_x}^2 + F_{prop_y}^2}$$

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

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

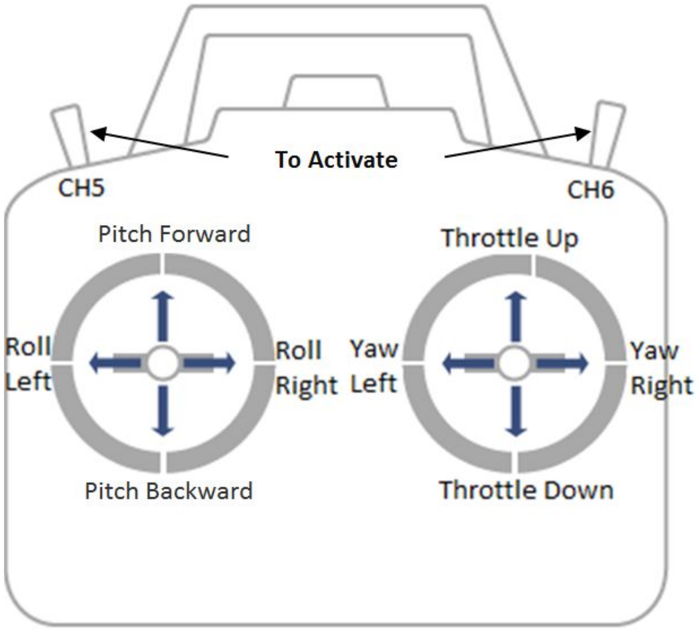
$$F_x = F_{prop_x} +/- Drag_x$$

$$F_y = F_{prop_y} +/- Drag_y$$

$$F_z = F_{prop_z} - mg +/- Drag_z$$

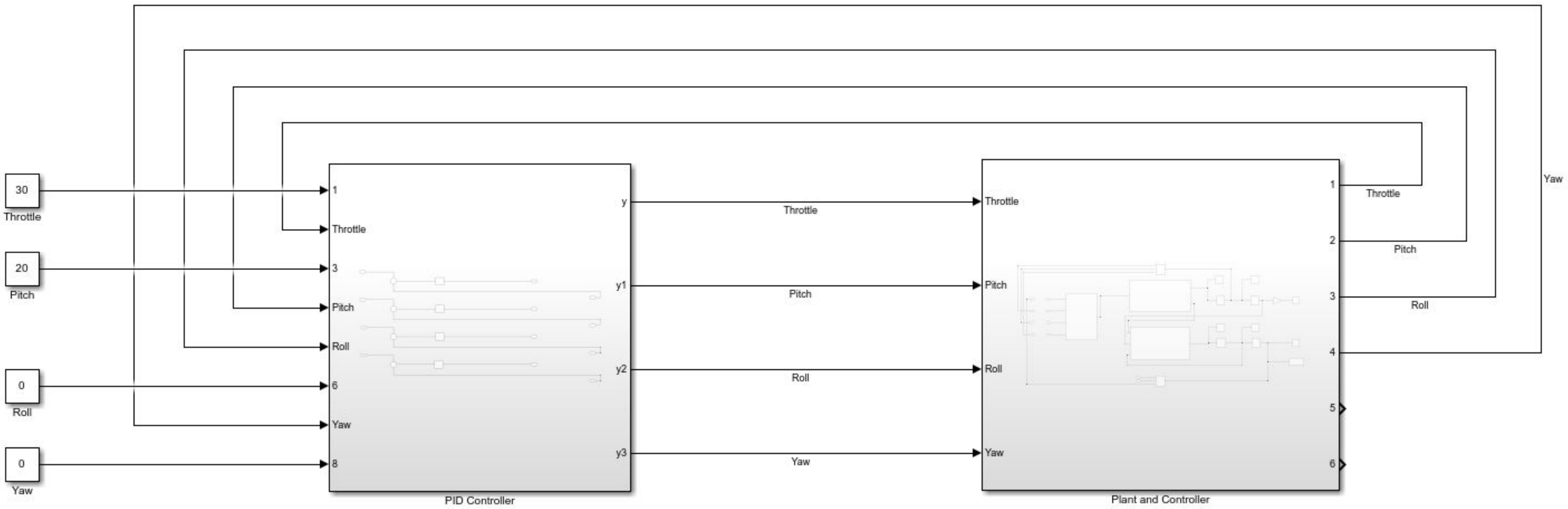
Model Introduction

- 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.

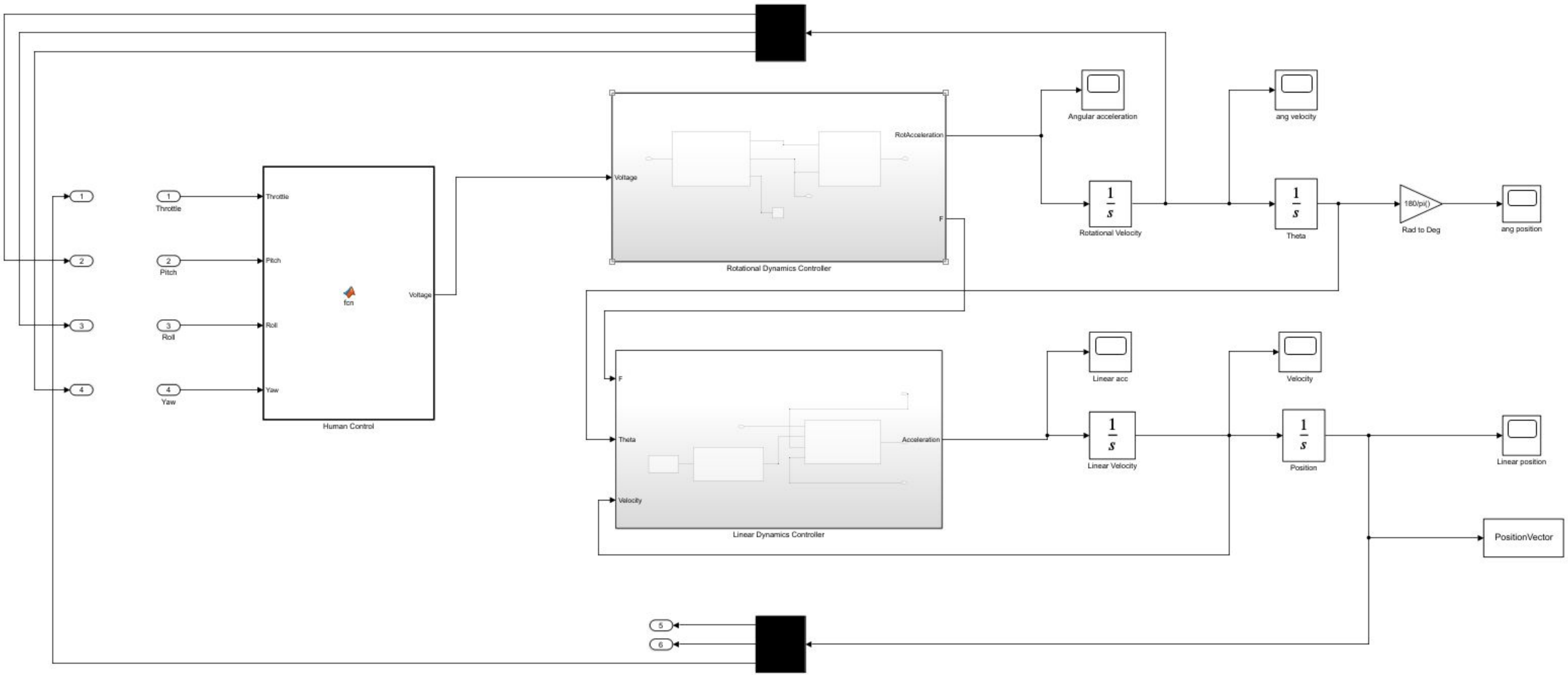


Remote Controller

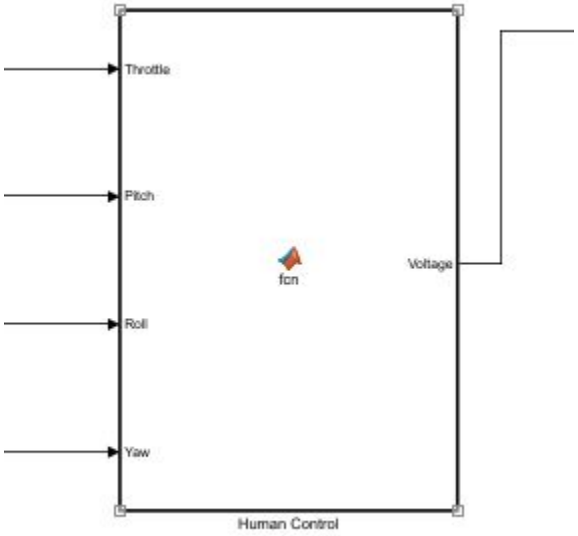
Overall Matlab/Simulink Model Design



Plant Subsystem



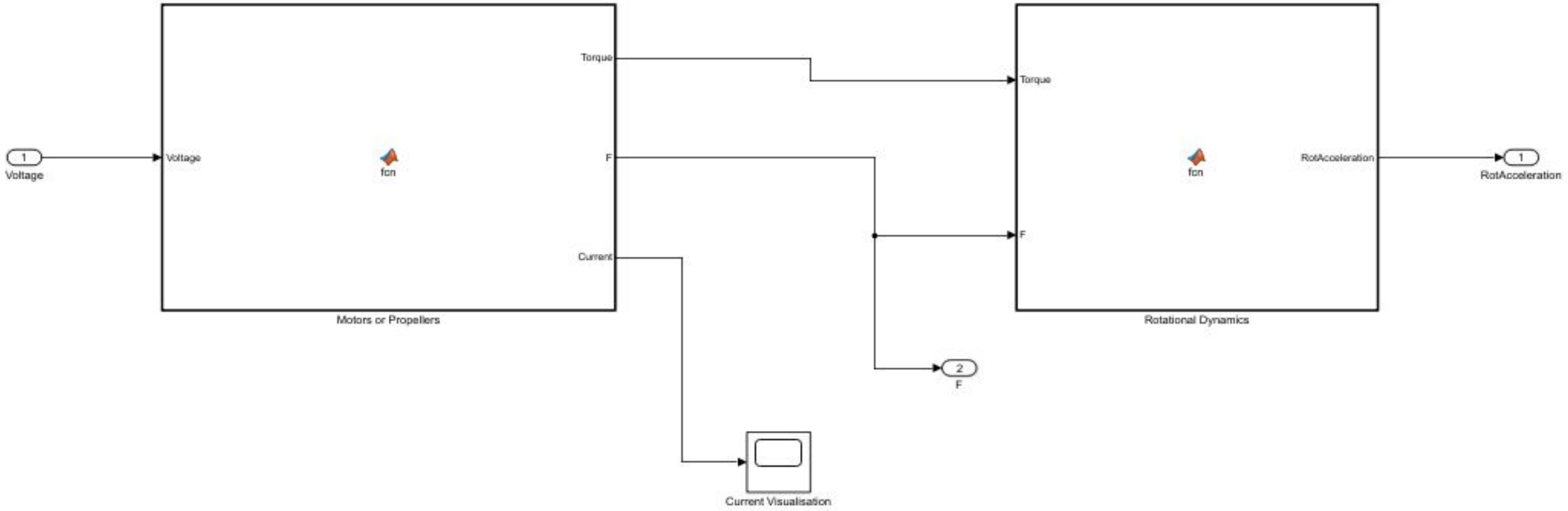
Limiting the Drone Power



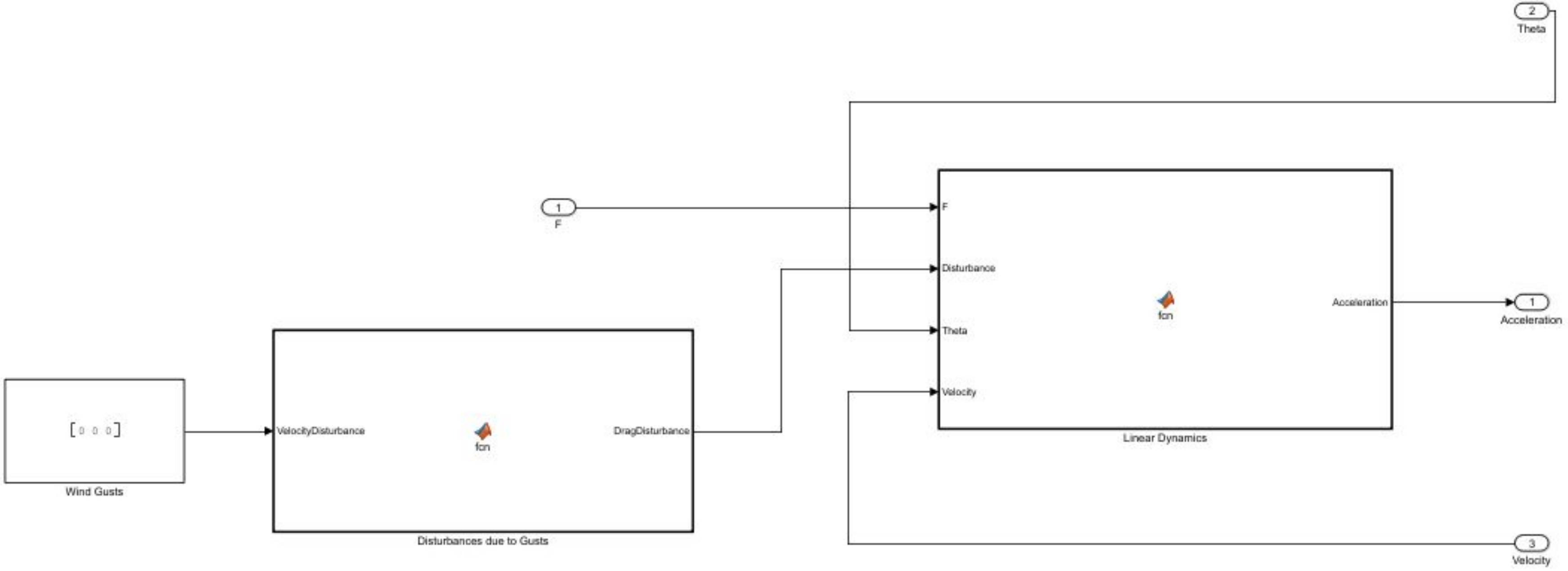
Plant and Controller/Human Control

```matlab
function Voltage = fcn(Throttle, Pitch, Roll, Yaw)
Correct = 0;
if (Pitch/2 + Roll/2 + Yaw/2) > (100 - Throttle)
    Correct = ((Pitch/2 + Roll/2 + Yaw/2) - (100 - Throttle));
end

if (Pitch/2 + Roll/2 + Yaw/2) > (Throttle)
    if Correct < ((Pitch/2 + Roll/2 + Yaw/2) - (Throttle))
       Correct = ((Pitch/2 + Roll/2 + Yaw/2) - (Throttle));
    end
end

if Correct ~= 0
    Pitch = Pitch - Correct/3*2;
    Roll = Roll - Correct/3*2;
    Yaw = Yaw - Correct/3*2;
end

Voltage=[(Throttle - Pitch/2 - Roll/2 - Yaw/2)*11.4/100
    (Throttle - Pitch/2 + Roll/2 + Yaw/2)*11.4/100
    (Throttle + Pitch/2 + Roll/2 - Yaw/2)*11.4/100
    (Throttle + Pitch/2 - Roll/2 + Yaw/2)*11.4/100
];
```

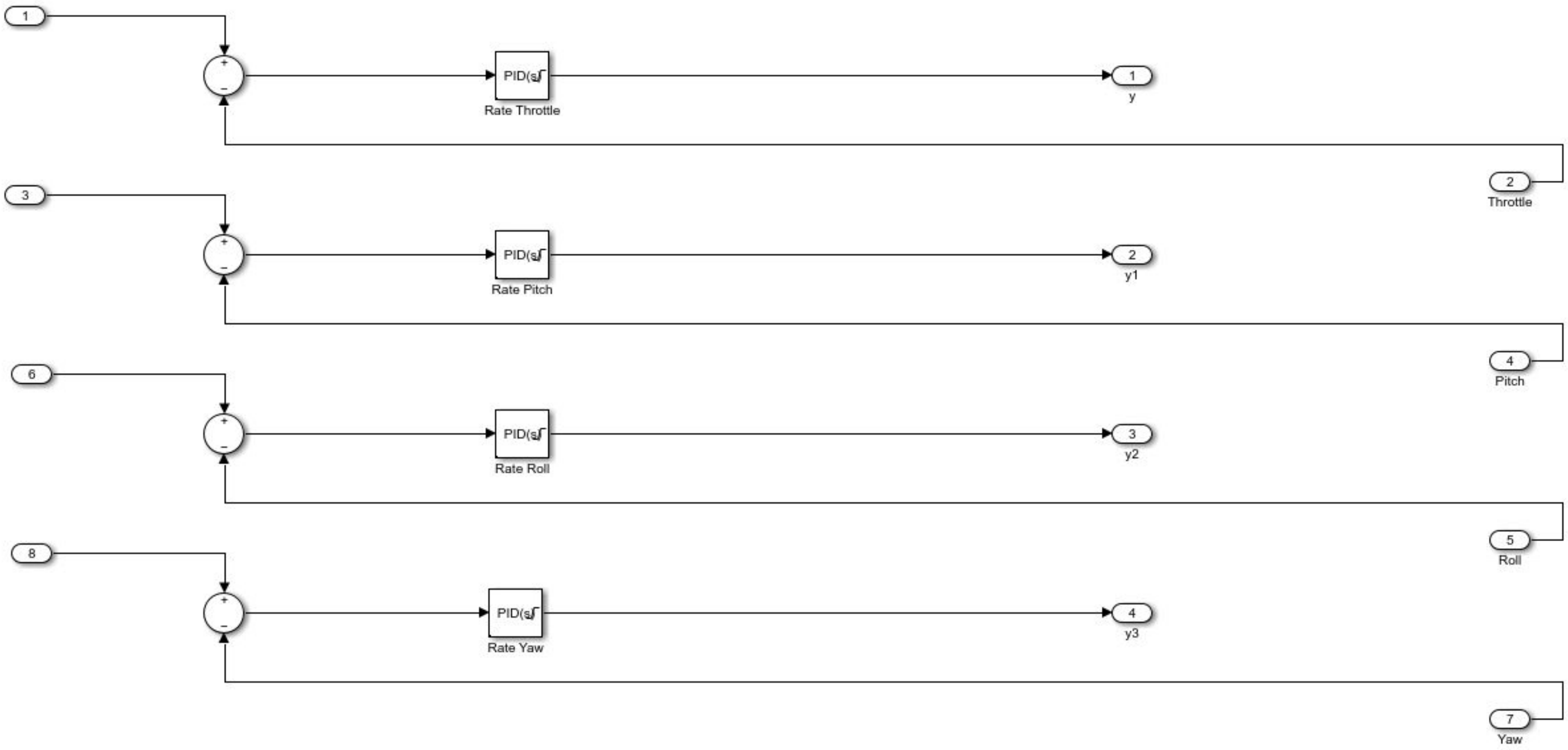
Rotational Dynamics subsystem



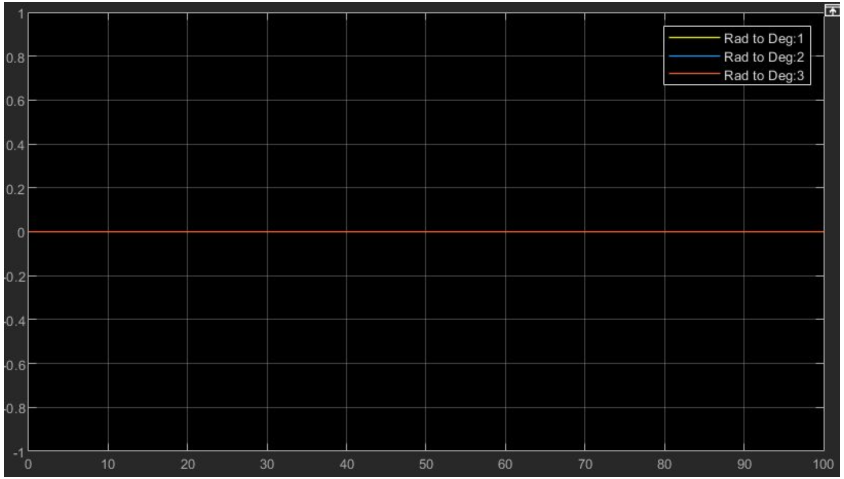
Linear Dynamics Subsystem



PID Controller Design

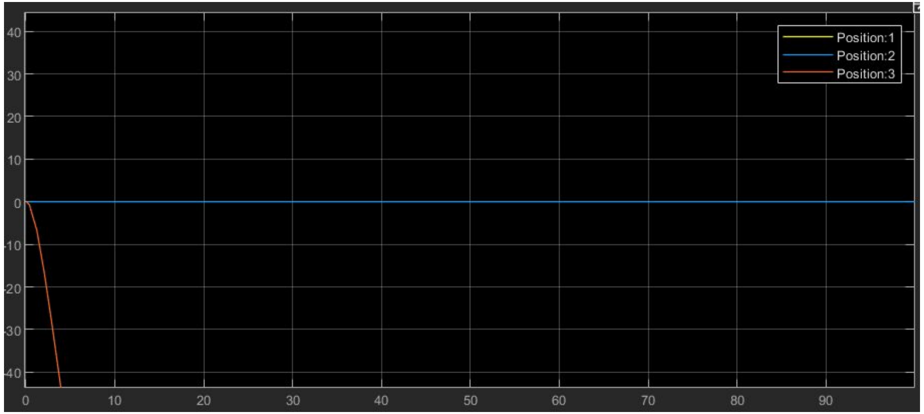


Drone at hover state



Terminology:

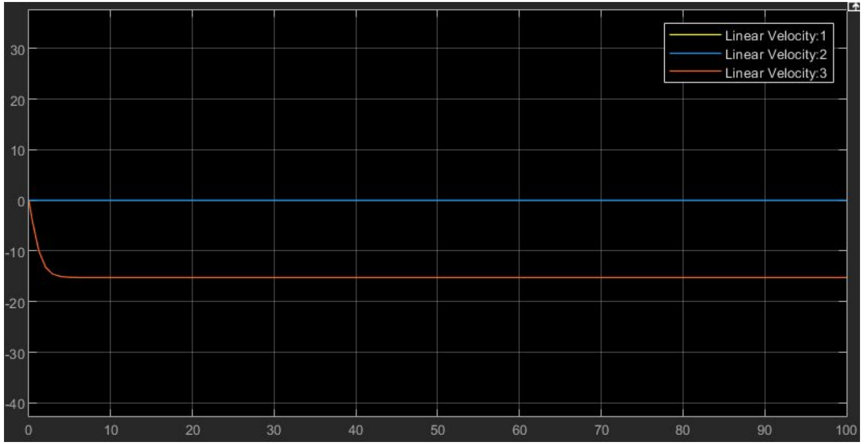
Yellow: For x axis
Blue: For y axis
Red: For z axis



No angular velocity in x, y or z direction.

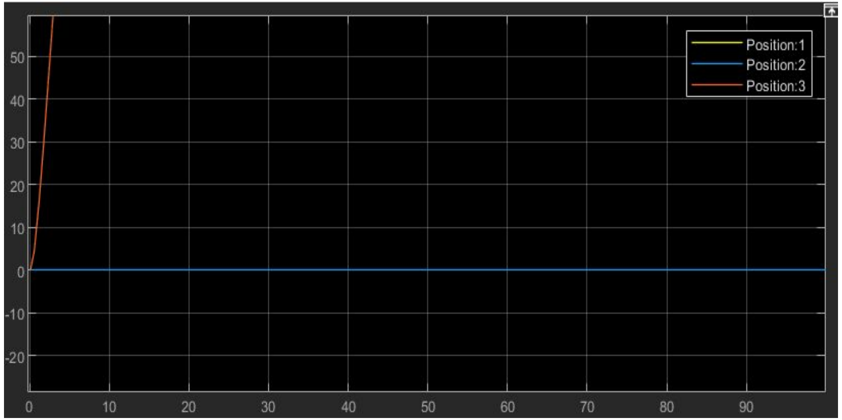
Drone will fall down due to it's own weight.

Drone at hover state

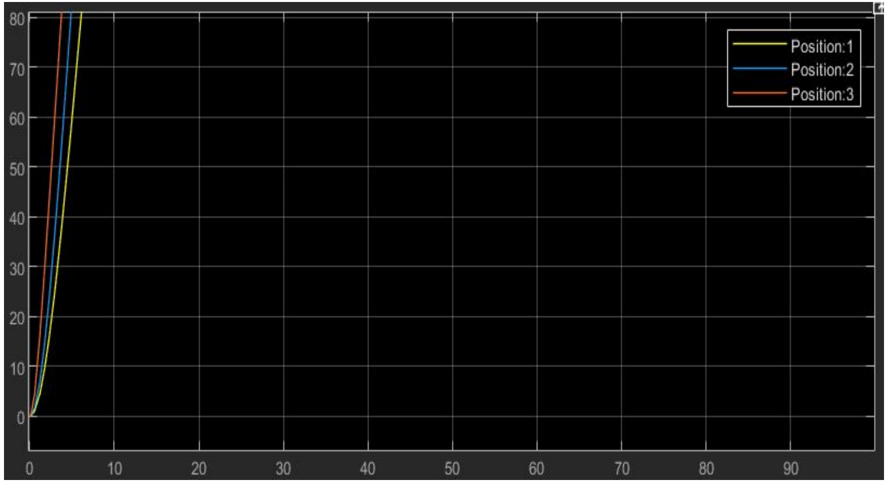


Due to air drag, the drone will eventually reach terminal velocity after sometime.

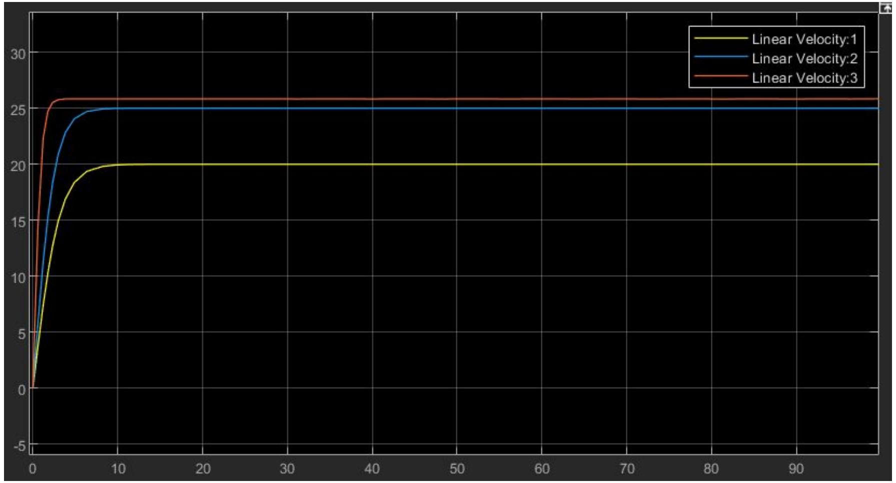
On adding **wind disturbance** in +z direction, drone moves upwards besides the force of gravity (without using PID controller)



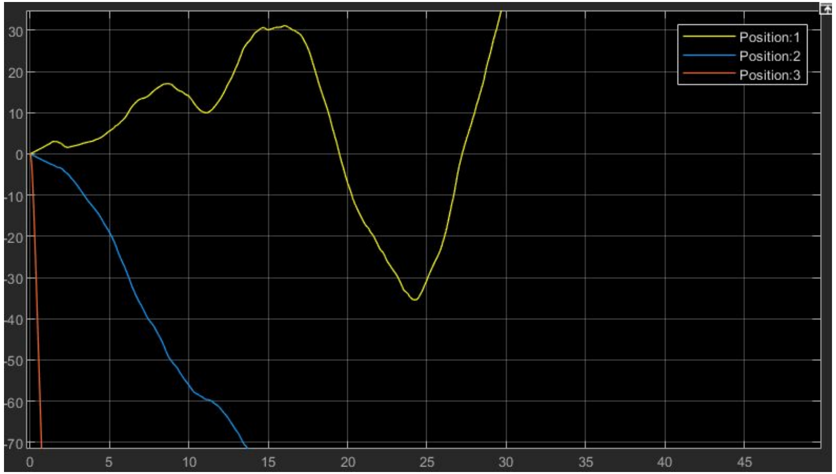
Drone at hover state



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.



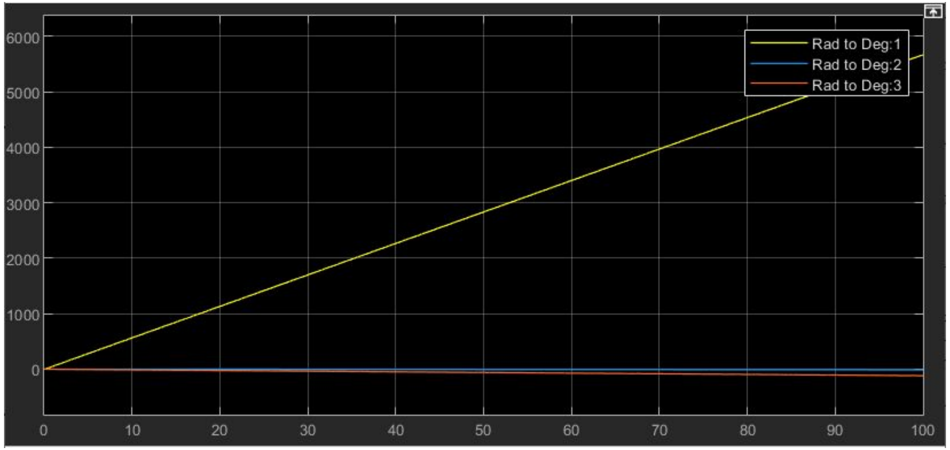
Pitch with Throttle



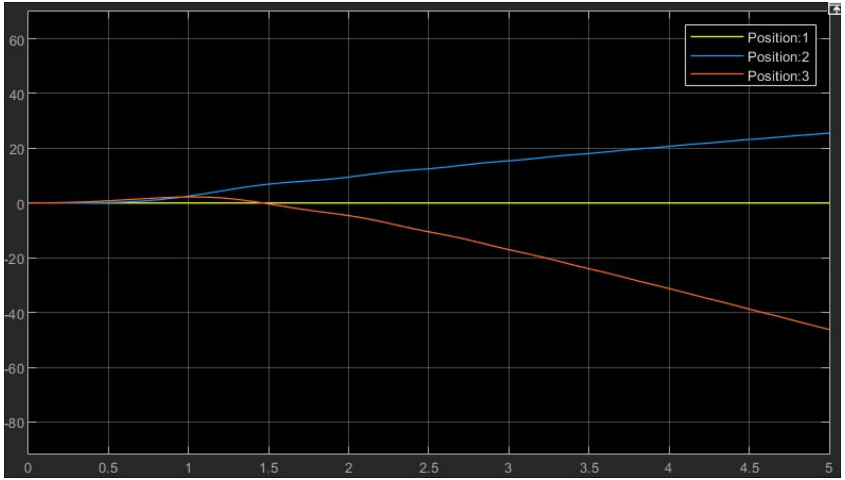
Throttle: 50, Pitch: 1
Roll: 0, Yaw: 0

Rotation about x axis

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

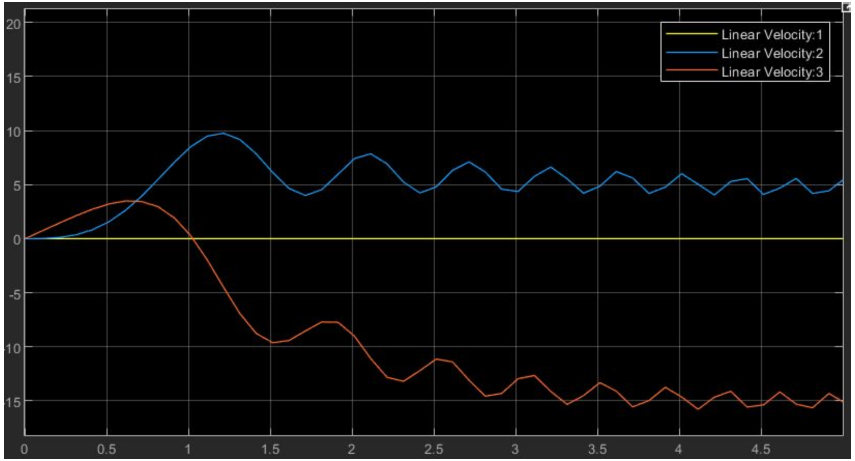


Pitch with Throttle



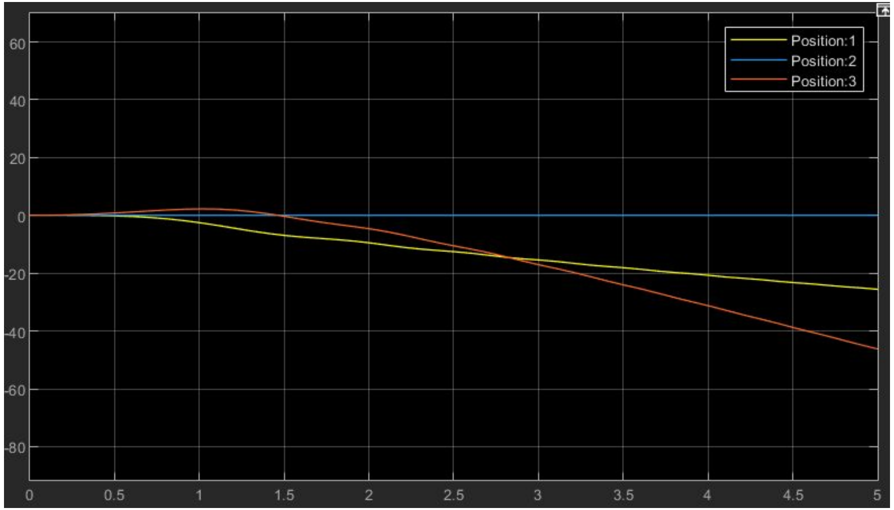
Throttle: 50, Pitch: 1
Roll: 0, Yaw: 0

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.



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.

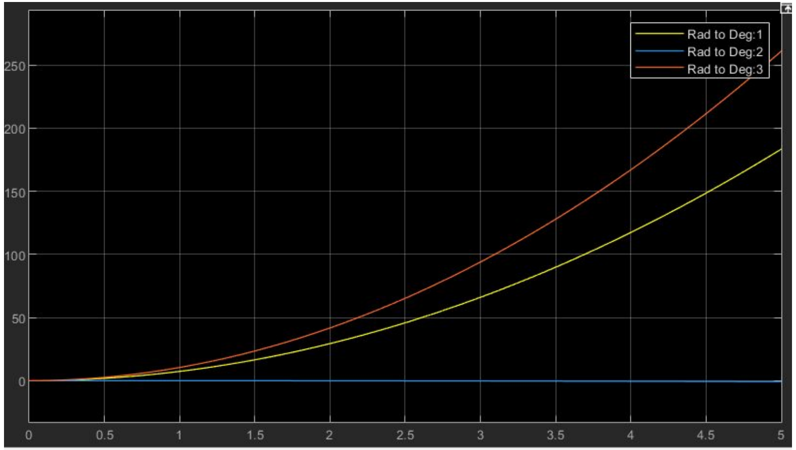
Pitch with Throttle and Yaw



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

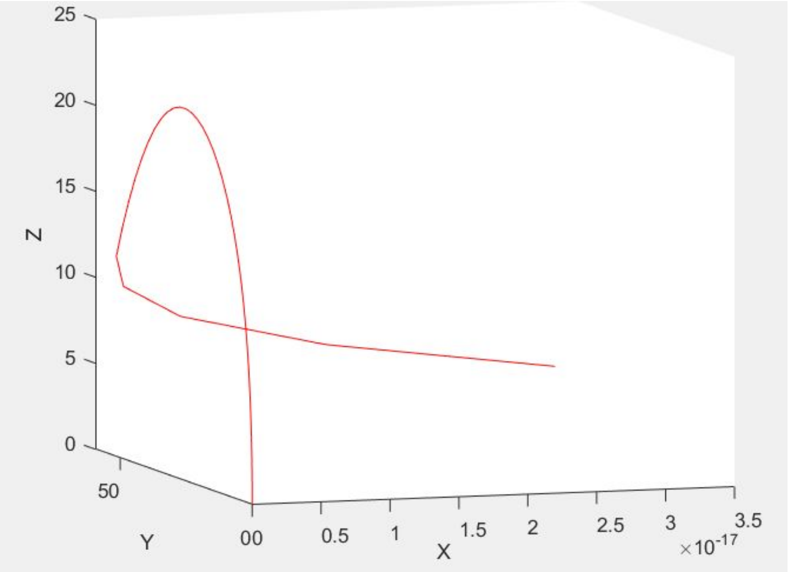
Throttle: 50, Pitch: 0
Roll: 0.5, Yaw: 0

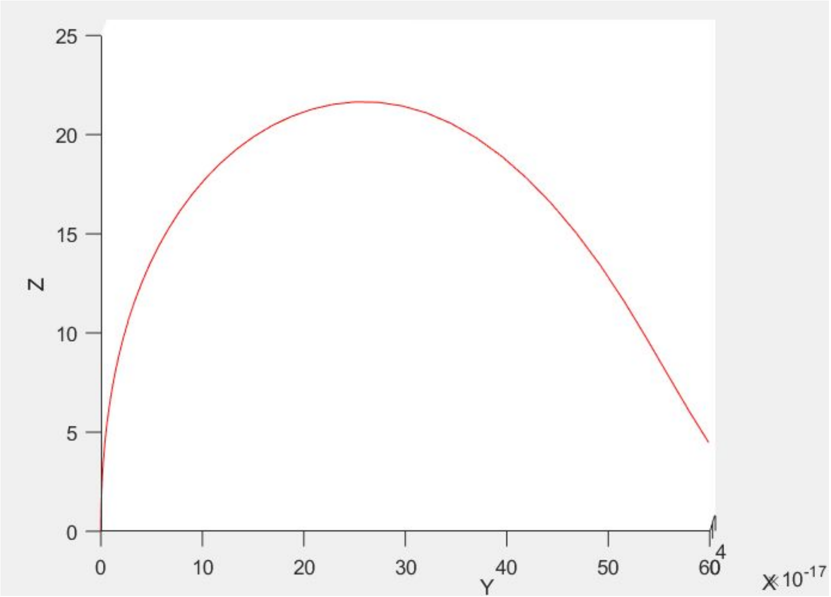
When rotating about y axis, it will first move in +z direction then after each 180 degrees interval, direction is reversed. Similarly for x axis also.



3D Visualization

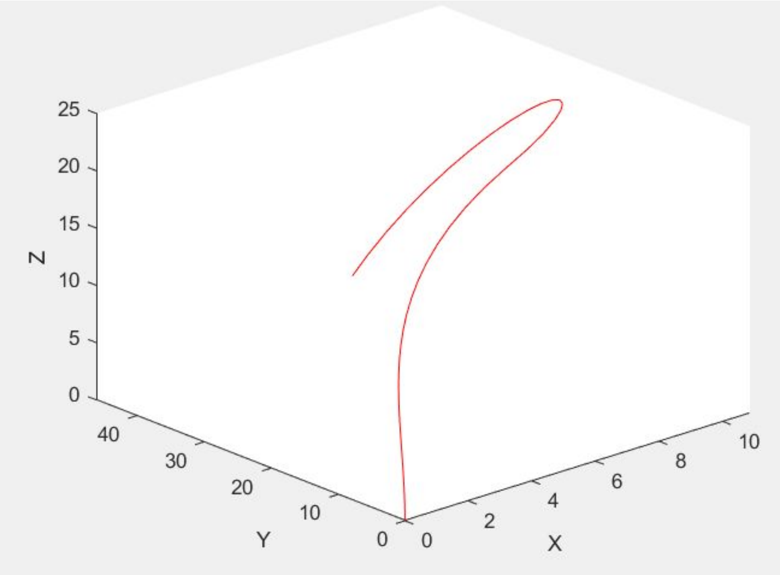
Throttle: 50, Pitch: 0.035
Roll: 0, Yaw: 0





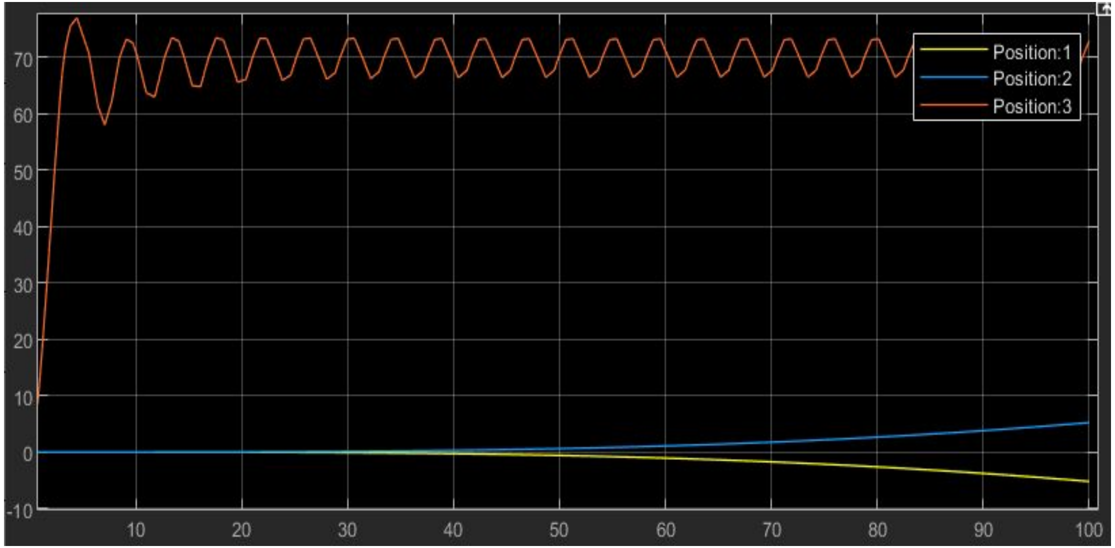

```
>> plot3(PositionVector(:,1), PositionVector(:,2), PositionVector(:,3), '-r')
```

3D Visualization



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

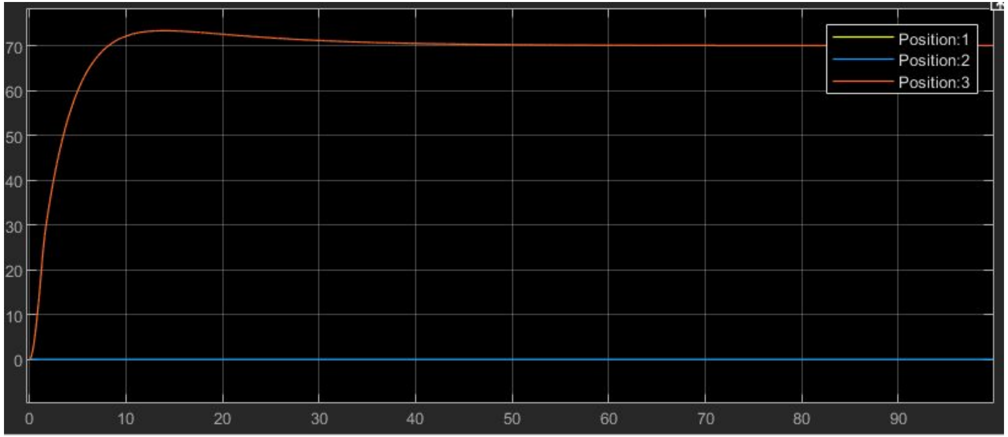
Altitude Control of Drone



Using PID Controller without tuning:

T_P= 5 (proportional gain)
T_I= 1 (integral gain)
T_D= 0 (derivative gain)

Altitude Control of Drone



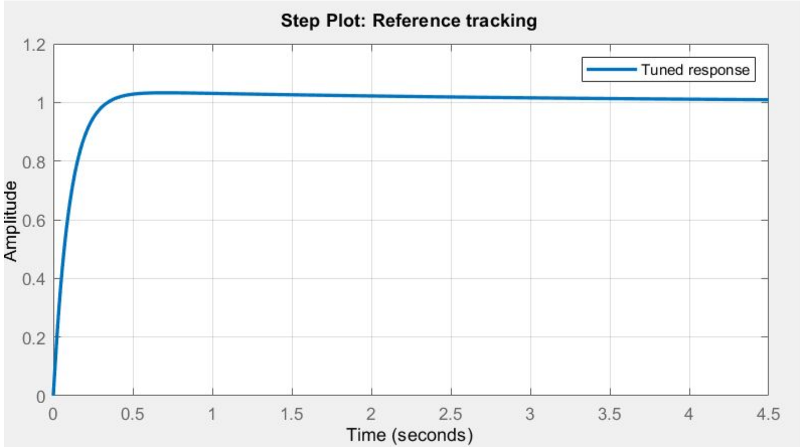
Controller Parameters: P = 13.2, I = 0.8918, D = 34.38, N = 1143

To increase speed of response, increase proportional gain.

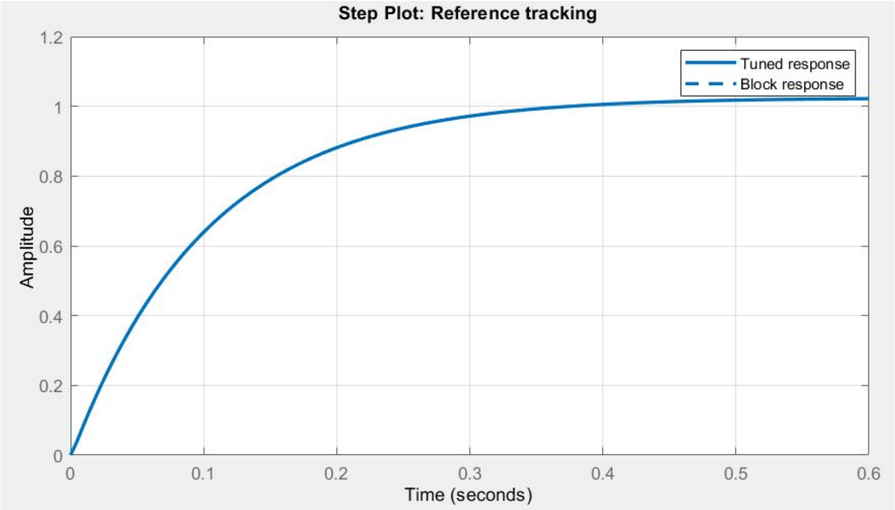
Tuning with PID controller

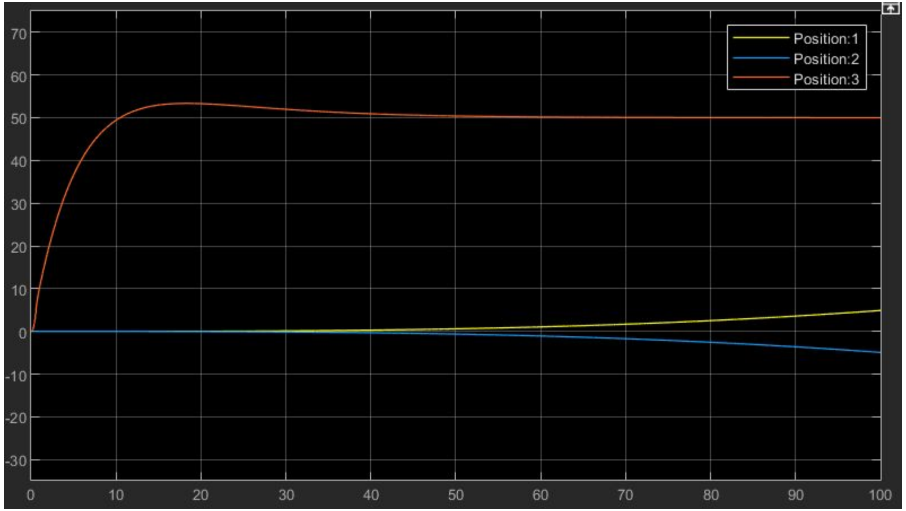
Throttle = 70.

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



Altitude Control with PID Controller

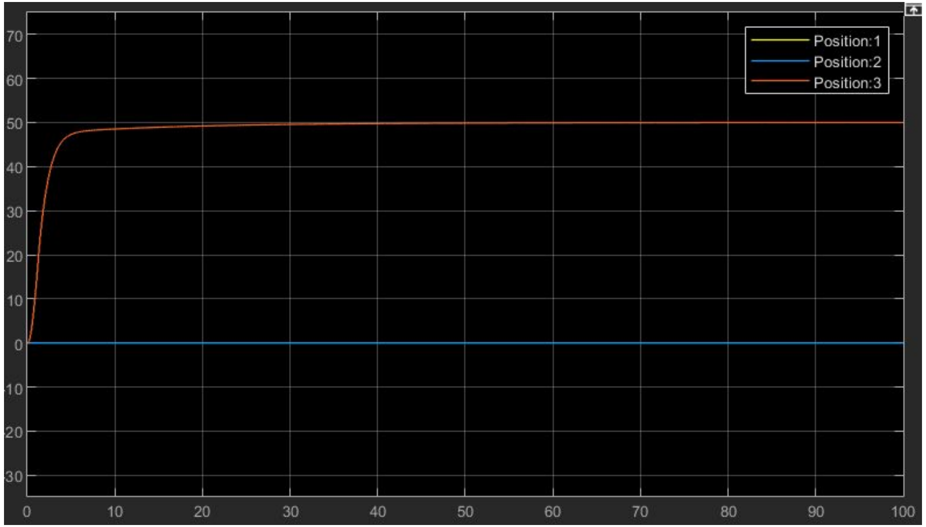


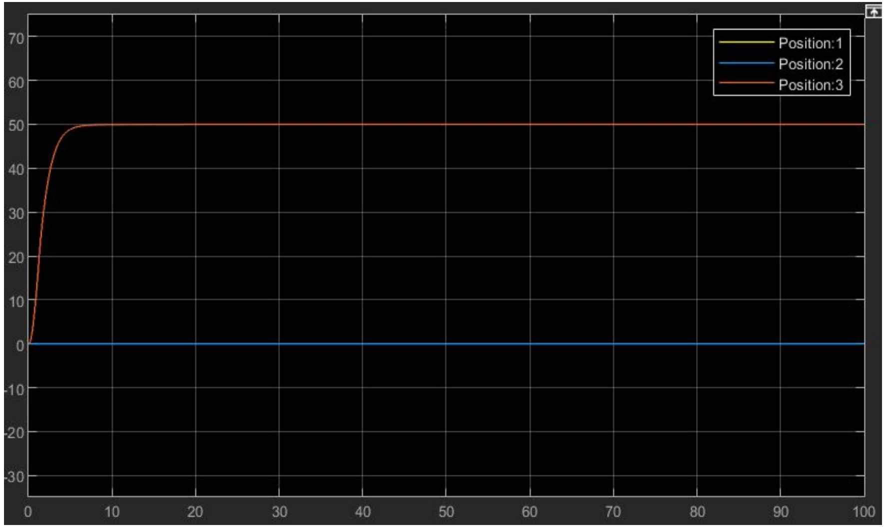


Tuning PID controller while trying not to overshoot.

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

Altitude Control with PID Controller





Proportional (P): 9.03103662111541

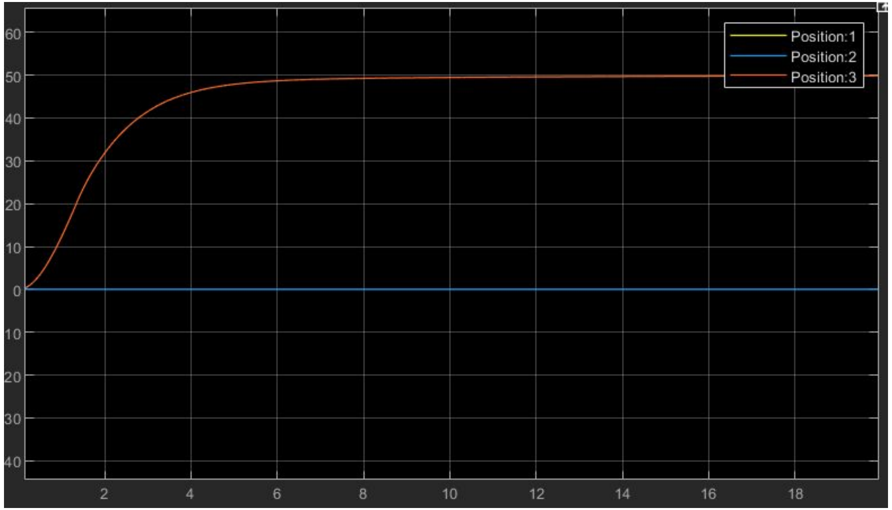
Integral (I): 0.52686425472253

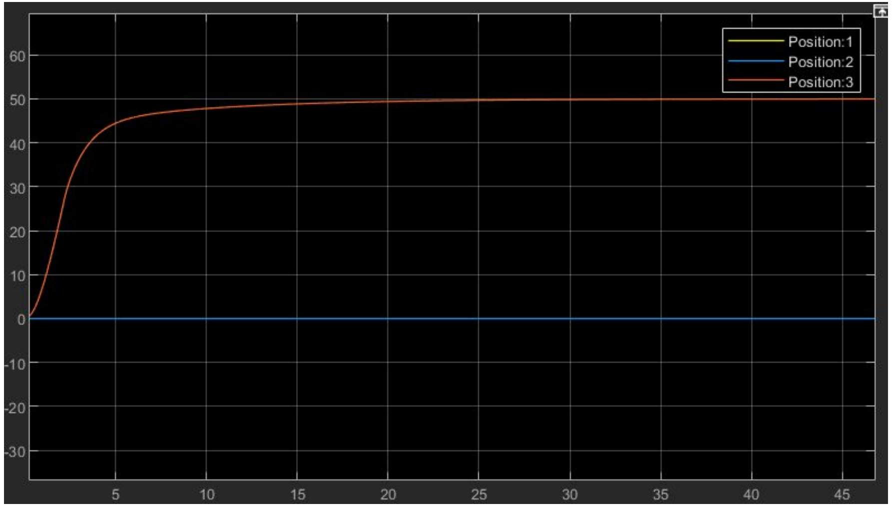
Derivative (D): 10

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



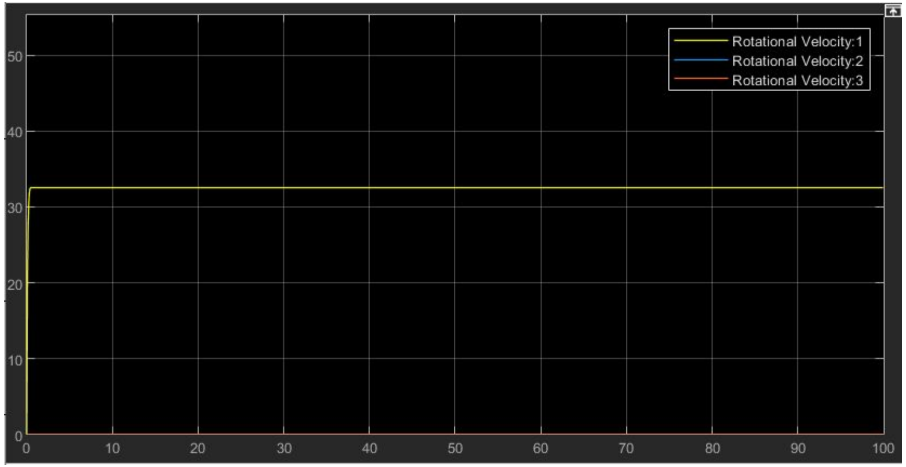


Wind disturbance in +z: -10 m/s
Taking around 8 seconds to reach

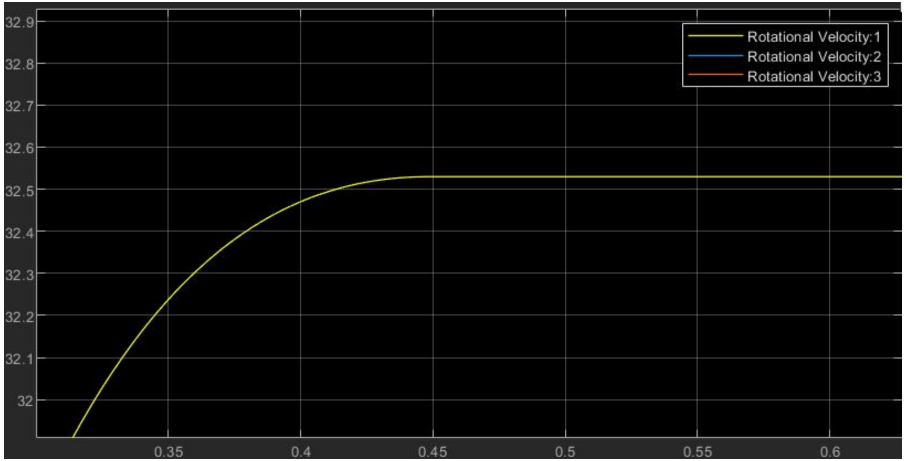
Wind disturbance in +z: -20 m/s
Taking about 15 seconds to reach

Pitch PID Controller

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

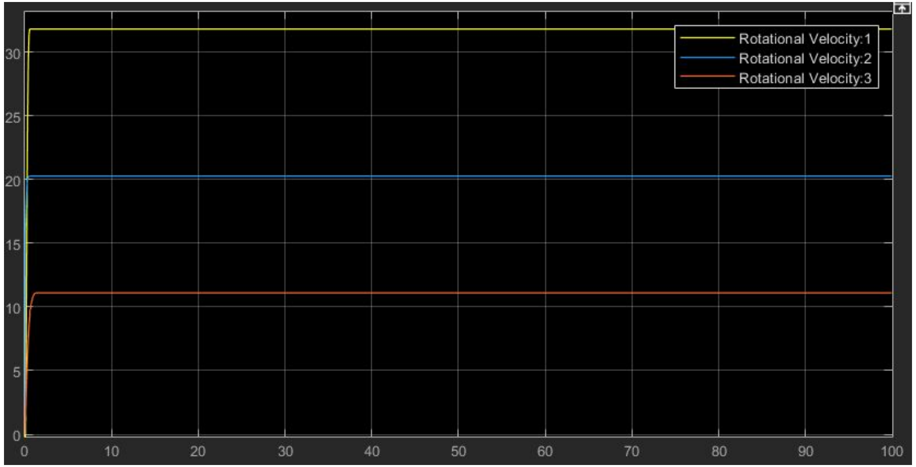


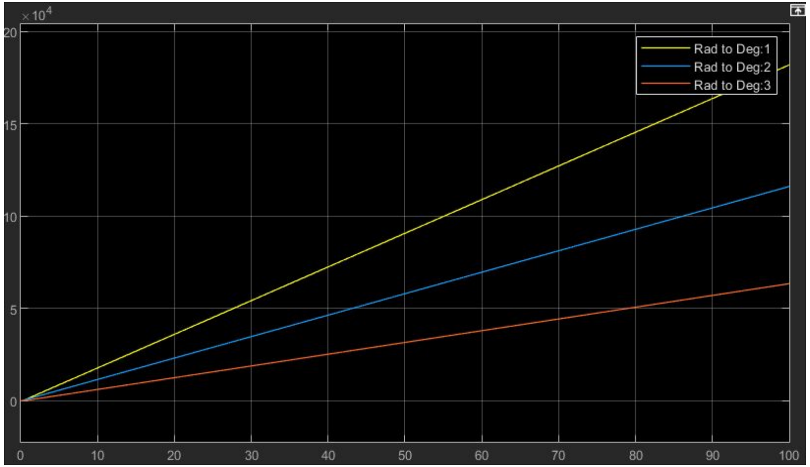
Throttle: 50, Pitch: 32.5, Roll: 0, Yaw: 0



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

```
ArchiveEntry "CS591"
Description "UAV_Drone".

/*
    Air Drags and Air Disturbances are neglected.
    Motor or Propellers limits are also neglected.
*/
/*
    a = Sin(thetax)
    b = Cos(thetax)
    c = Sin(thetay)
    d = Cos(thetay)

    a' = b
    b' = -a
    c' = d
    d' = -c

    e = Sin(thetaz)
    f = Cos(thetaz)
    g = Sin(thetaXY)
    h = Cos(thetaXY)

    e' = f
    f' = -e
    g' = h
    h' = -g
*/
```

```

/* Represent all the constant terms used */

Definitions
    Real Ax, Ay, Az;
    Real m, g, Ix, Iy, Iz;
    Real rho, Cd;
End.

/* Represent all the state variables */

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

```

Formal Verification using KeYmaera X

```
Problem
    /* All the input values are given here */
    throttle = 30 & pitch = 0 & roll = 0 & yaw = 0 & Ix = 0.003 & Iy = 0.003 & Iz = 0.007 & m = 0.734 & g = 9.81 &
    Ax = 0.0197 & Ay = 0.0197 & Az = 0.0512 & rho = 1.225 & Cd = 1
    ->
    [{
        /* Represent the continuous part of dynamics */
        {
            x' = vx, y' = vy, throttle' = vz, vx' = Accx, vy' = Accy, vz' = Accz,
            thetax' = pitch, thetay' = roll, thetaz' = yaw, pitch' = RotAccx, roll' = RotAccy, yaw' = RotAccy,
            a' = b, b' = -1*a, c' = d, d' = -1*c,
            e' = f, f' = -1*e, g' = h, h' = -1*g,
            ThetaXY' = -1 / (1 + (Fprop1/Fprop2)^2)
        }
        /* Represent the discrete part of dynamics */
        {
            /* Motors or Propellers */
            voltage1 := (throttle - pitch/2 - roll/2 - yaw/2)*11.4/100;
            voltage2 := (throttle - pitch/2 + roll/2 + yaw/2)*11.4/100;
            voltage3 := (throttle + pitch/2 + roll/2 - yaw/2)*11.4/100;
            voltage4 := (throttle + pitch/2 - roll/2 + yaw/2)*11.4/100;

            RPM1 := -2.6931*(Voltage1^3) + 1400*Voltage1;
            RPM2 := -2.6931*(Voltage2^3) + 1400*Voltage2;
            RPM3 := -2.6931*(Voltage3^3) + 1400*Voltage3;
            RPM4 := -2.6931*(Voltage4^3) + 1400*Voltage4;

            F1 := (2*(10^-15)*(RPM1^3) - 4*(10^-11)*(RPM1^2) + 3*(10^-7)*RPM1 + 0.1013)*1.225*((RPM1/60)^2)*0.0016;
            F2 := (2*(10^-15)*(RPM2^3) - 4*(10^-11)*(RPM2^2) + 3*(10^-7)*RPM2 + 0.1013)*1.225*((RPM2/60)^2)*0.0016;
            F3 := (2*(10^-15)*(RPM3^3) - 4*(10^-11)*(RPM3^2) + 3*(10^-7)*RPM3 + 0.1013)*1.225*((RPM3/60)^2)*0.0016;
            F4 := (2*(10^-15)*(RPM4^3) - 4*(10^-11)*(RPM4^2) + 3*(10^-7)*RPM4 + 0.1013)*1.225*((RPM4/60)^2)*0.0016;

            Torque1 := 4*(10^-14)*(RPM1^3) + 8*(10^-12)*(RPM1^2) + 3*(10^-6)*RPM1;
            Torque2 := 4*(10^-14)*(RPM2^3) + 8*(10^-12)*(RPM2^2) + 3*(10^-6)*RPM2;
            Torque3 := 4*(10^-14)*(RPM3^3) + 8*(10^-12)*(RPM3^2) + 3*(10^-6)*RPM3;
            Torque4 := 4*(10^-14)*(RPM4^3) + 8*(10^-12)*(RPM4^2) + 3*(10^-6)*RPM4;
```

Formal Verification using KeYmaera X

```
        /* Rotational Dynamics */
        Moment1 := (F3 + F4)*0.237/2 - (F1 + F2)*0.237/2;
        Moment2 := (F3 + F2)*0.237/2 - (F1 + F4)*0.237/2;
        Moment3 := Torque4 - Torque1 + Torque2 - Torque3;

        RotAccx := Momentx/Ix;
        RotAccy := Momenty/Iy;
        RotAccz := Momentz/Iz;

        /* Linear Dynamics */
        Fprop1 := c * b * (F1 + F2 + F3 + F4);
        Fprop2 := a * d * (F1 + F2 + F3 + F4);
        Fprop3 := b * d * (F1 + F2 + F3 + F4);

        XY2D := sqrt(Fprop1^2 + Fprop2^2);

        ?Fprop3 >= 0; Fprop1 := XY2D * (e*h + f*g); ++
        ?Fprop3 >= 0; Fprop2 := XY2D * (f*h - e*g); ++
        ?Fprop3 < 0; Fprop1 := XY2D * (g*f - e*h); ++
        ?Fprop3 < 0; Fprop2 := XY2D * (e*g + f*h);

        Forcex := F1;
        Forcey := F2;
        Forcez := F3 - m * g;

        Accx := Forcex / m;
        Accy := Forcey / m;
        Accz := Forcez / m;
      }
    }*@invariant(throttle>=25 & throttle<=35)] throttle>=25 & throttle<=35
End.

Tactic "UAV_Drone: automatic"
  auto
End.

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