Drone or UAV Controller

A CS591 Course Project
Term Paper Submission

by

Samay Varshney
(180101097)

under the guidance of

Dr. Purandar Bhaduri

to the

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
GUWAHATI - 781039, ASSAM

CERTIFICATE

*This is to certify that the work contained in this thesis entitled "**Drone or UAV Controller**" is a bonafide work of (**Roll No. 180101097**), carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati under my supervision and that it has not been submitted elsewhere for a degree.*

Supervisor: **Dr. Purandar Bhaduri**

Professor

Department of Computer Science & Engineering

Indian Institute of Technology Guwahati, Assam

Acknowledgements

I would like to express my gratitude to my supervisor **Dr. Purandar Bhaduri** for giving me the opportunity to explore this new field of Cyber Physical Systems and the much needed zest to delve into some of the state-of-the-art works.

Contents

Chapter 1

Abstract & Introduction

1.1 Abstract

Cyber-Physical Systems are safety-critical real-time reactive systems which may inflict serious consequences upon erroneous behaviour. This motivates us to explore them. Here, we explored the **UAV** (Unmanned Aerial Vehicles) or **Drones** as a cyber physical system with the property to maintain the altitude, position and rotation of the drone at different levels in the environment. We modeled and simulated the basic features of UAV or Drone using MATLAB/Simulink and verified using the **KeYmaera X** tool, a tool used to verify the hybrid systems.

1.2 Problem Statement

Maintain the altitude, rotation about x (pitch), y (roll) and z (yaw) axis and position of the drone on different inputs values and tune them using PID controller. The remote control of the drone as shown in 1.1 will be having inputs as throttle, pitch, roll and yaw. The user will use the remote control for flying the drone at different levels according to his/her needs. Thrust, Roll, Pitch and Yaw can be commanded independently. The overall system will also face air drags and air disturbances due to the environment and the motors will limited

power of how much current they can provide to the drone for movement.

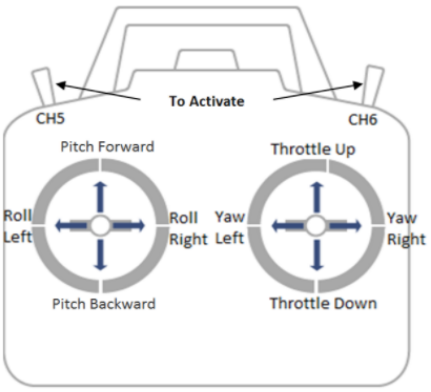


Fig. 1.1: Remote Controller

1.3 Introduction

Drones plays an important role in real life. They are used in Aerial Photography & Videography, Disaster Relief, Product Delivery, Search and Rescue and in many other wide variety of applications.

UAV or Drone as a cyber physical system consists of three main components: environment, plant and controller. Environment is the outside world and controls the input values and the wind velocity leading to air drags and wind disturbances. Plant contains different sub components such as Human Control, Linear Dynamics, Rotational Dynamics, Motors, Disturbances. Controller is the main component which controls the throttle, pitch, roll and yaw of the drone.

The characteristics of the drone used are:

- Diagonal length: 335 mm
- Fontal Area: 0.0197 m^2
- Battery Capacity: 3830 mAh
- Max Discharge Current: 77 A
- Weight: 0.743 kg
- Motor KV: 1400 rpm/V
- Voltage: 11.4 V
- Propeller Size: 8x4 inches

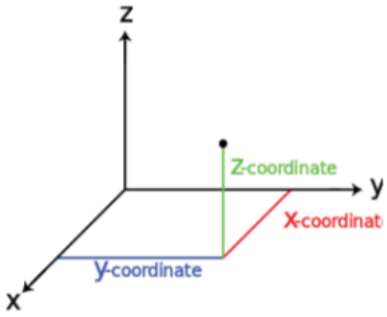
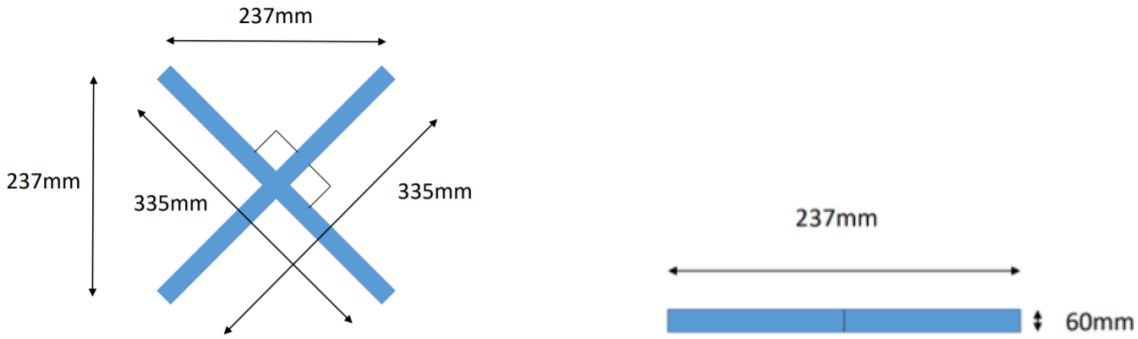


Fig. 1.2: X, Y and Z Axis Conventions

1.3.1 Motor Limitations

Apart from this, maximum operating voltage will be 11.4 Volts to consider an average over one battery discharge and a max current of 77 Amperes. 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.



(a) Top View (b) Side View

Fig. 1.3: Shape of Drone

1.3.2 RPM Motor Constant KV

The APC, a company which manufactures propellers has published the detailed performance of its propellers on their website. We used 8x4 propeller data values for extracting different formulas for this drone dynamics by fitting polynomial equations. One example is shown in

Fig 1.4.

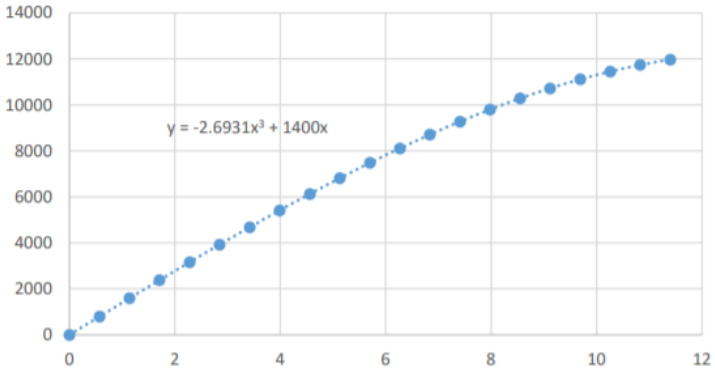


Fig. 1.4: RPM against Voltage with Torque Loading

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.

1.4 Organization of The Term Paper

This chapter introduces the problem statement covered in this term paper. we provided a brief overview of the topic and talked about the importance of the drones. The rest of the chapters are arranged as follows: In next chapter we discuss all the preliminaries that would help brush up the keywords required. In **Chapter 3**, we modelled the system using MATLAB/Simulink. In **Chapter 4**, we simulate the model using MATLAB/Simulink under different values of throttle, pitch, roll and yaw. In **Chapter 5**, we verified the hybrid model using KeYmaera X tool. And finally in **Chapter 6**, we will conclude with the final remark.

Chapter 2

Preliminaries

The following section contains a background of the different terminologies that will be used in the later sections.

2.1 Hybrid System

Hybrid system is the dynamical system that exhibits both continuous and discrete dynamic behaviour. A system that can both flow described by a differential equation and jump described by a state machine. Here, we represent the UAV or drone as a hybrid system.

2.2 Forces at Play

Fig. 2.1: Shape of Drone

2.3 Functionality of Drone

The RPM (Revolutions per minute) is used to derive Voltage given to each of the four propellers of the drone. Using this, Torque, Thrust produced by each propeller can be calculated. Further, moments and forces in x, y and z directions can be calculated using the basic physics formulas.

The torques necessary to rotate each propellers also act on the drone. Propellers 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. In Figure 2.2, F1, F2, F3 and

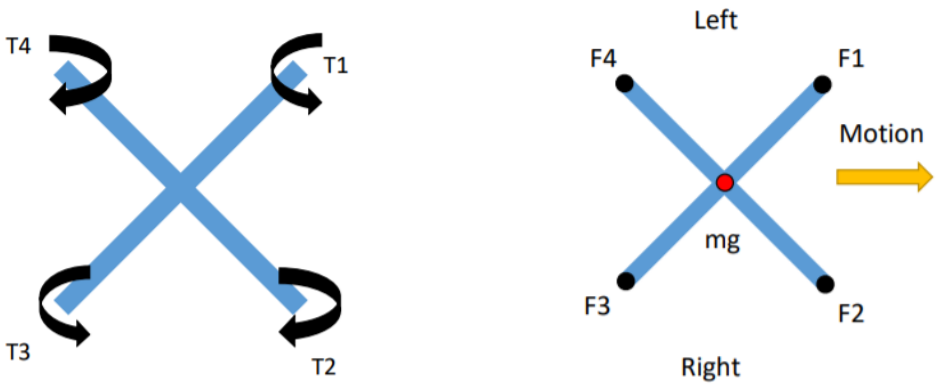


Fig. 2.2: Motion and Forces

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.

Chapter 3

Model

So far, we have looked at the basic terminologies that would be used in the rest of the paper. We now turn our attention towards the modeling of UAV or Drone.

3.1 Overall Model System

Fig 3.1 follows the model of the cyber physical system.

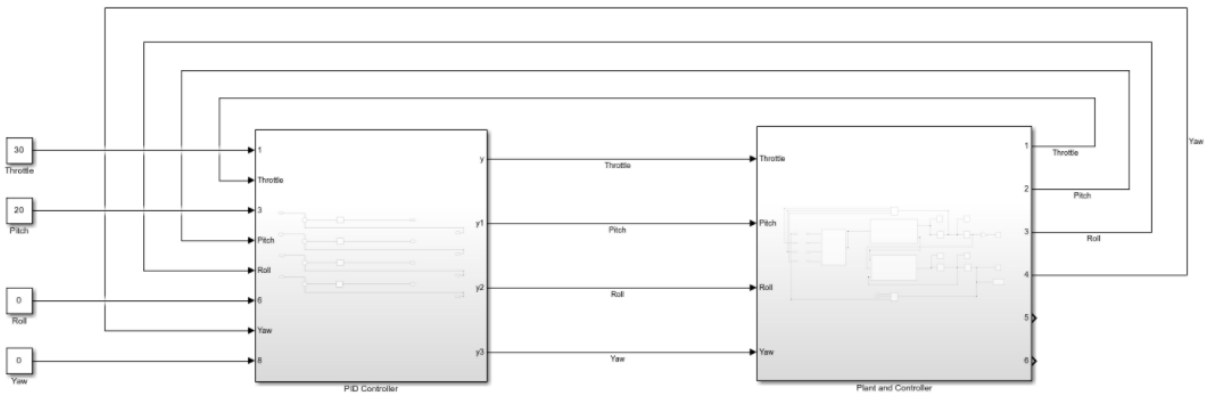


Fig. 3.1: Cyber Physical System

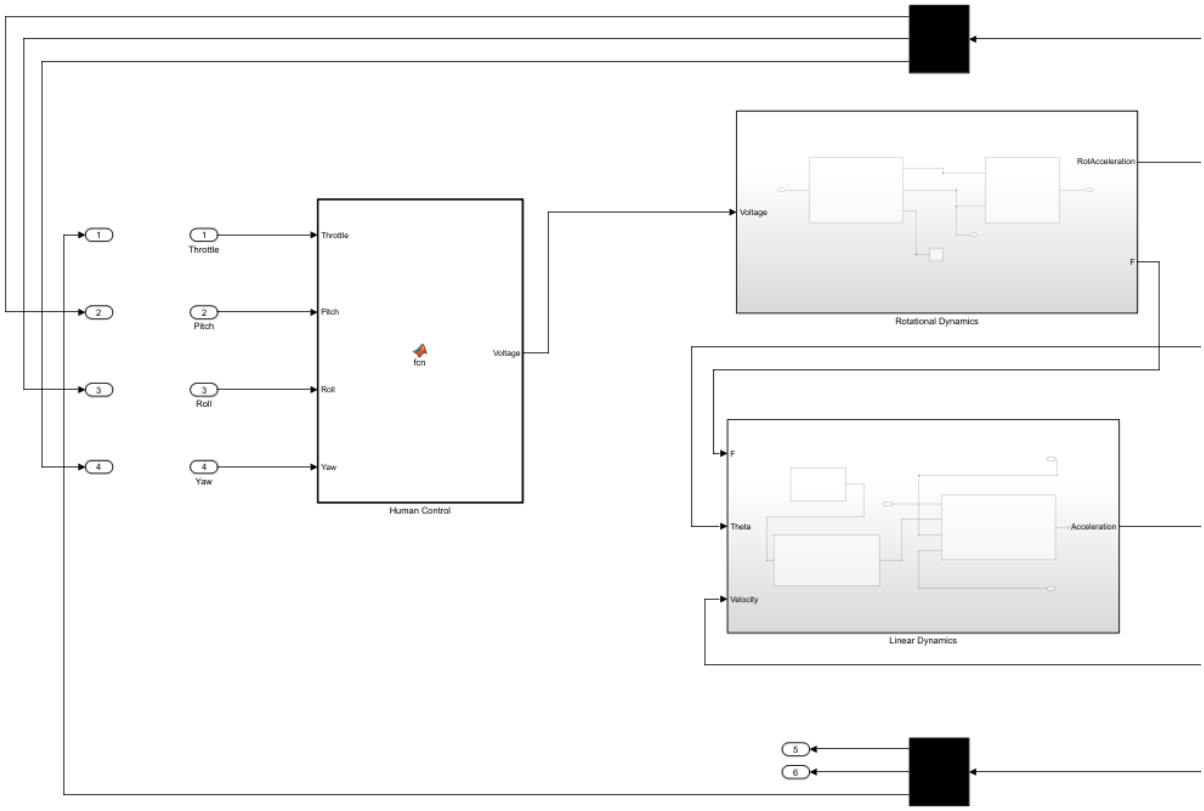


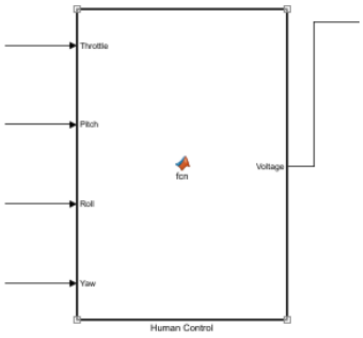
Fig. 3.2: Plant

3.2 Plant

Plant takes Throttle, Pitch, Roll and Yaw as inputs and gives linear and angular positions as final outputs as shown in Fig 3.2. It contains Human Control, Rotational Dynamics, Linear Dynamics blocks as it's sub-components.

3.2.1 Human Control

It takes all the human provided inputs and produces Voltage acting on each of the four propellers. This component also limits the power of drone on the high input values.



Plant and Controller/Human Control

```
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
];
```

Fig. 3.3: Human Control

3.2.2 Rotational Dynamics

Fig 3.4 represents the model of rotational dynamics of drone. It's first module, **Motors or Propellers**, takes Voltage as input and produces Torque, F and Current as outputs according to below mentioned equations for each propeller.

$$RPM_i = -2.6931 * Voltage_i{}^3 + 1400 * Voltage_i$$

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

$$F_i = C_t * 1.225 * (RPM_i/60)^2 * 0.2^4$$

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

$$Current_i = 1400 * Torque_i$$

Here $Voltage_i, Torque_i, Current_i$ represents Voltage, Torque, Current acting on ith propeller respectively.

Second module takes input as Torque and F and produces Rotational Acceleration as output. The moment of inertia about x, y and z axis can be calculated using the basic physics formulas.

$$I = [0.003, 0.003, 0.007]$$

$$Moment_x = (F_3 + F_4) * 0.237/2 - (F_1 + F_2) * 0.237/2$$

$$Moment_y = (F_3 + F_2) * 0.237/2 - (F_1 + F_4) * 0.237/2$$

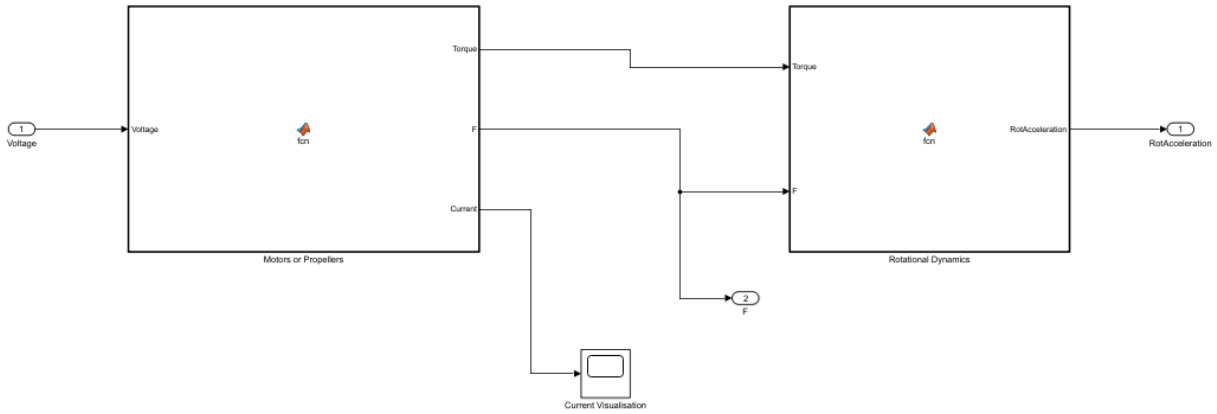


Fig. 3.4: Rotational Dynamics

$$Moment_z = Torque_4 - Torque_1 + Torque_2 - Torque_3$$

$$RotAcceleration_x = Moment_x / I_x$$

$$RotAcceleration_y = Moment_y / I_y$$

$$RotAcceleration_z = Moment_z / I_z$$

Here I is moment of inertia, $Moment_i$ is the moment of inertia and $RotAcceleration_i$ is the angular acceleration about ith axis.

3.2.3 Linear Dynamics

Fig 3.5 represents the model of linear dynamics of drone. It's first module, **Disturbances due to Gusts**, takes wind velocity disturbance as input and produces drag disturbance as output in x, y and z axis using the below equation.

$$DragDisturbance_i = 0.5 * \rho * VelocityDisturbance_i{}^2 * A_i * C_d$$

Here A is the area of drone in x, y and z directions, C_d is drag coefficient, ρ is the air density at sea level.

Second module takes F, drag disturbance, angular positions, linear velocity as input and produces linear acceleration as output.

$$F_{\text{prop}_\text{x}} = \sin(\theta_y) * \cos(\theta_x) * (F_1 + F_2 + F_3 + F_4)$$

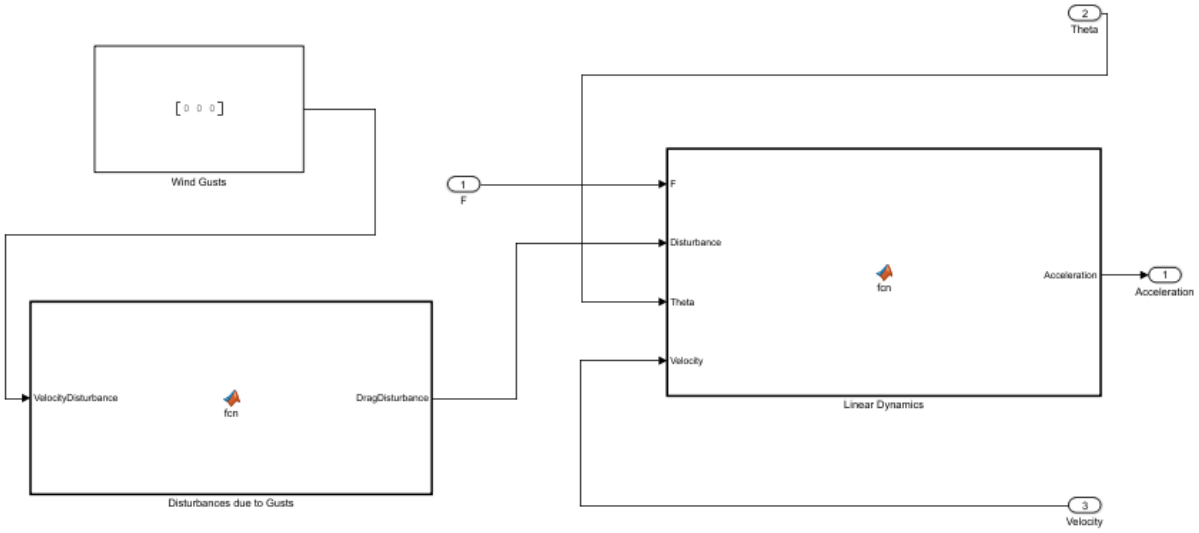


Fig. 3.5: Linear Dynamics

$$F_{\text{prop}_\text{y}} = \sin(\theta_x) * \cos(\theta_y) * (F_1 + F_2 + F_3 + F_4)$$

$$F_{\text{prop}_\text{z}} = \cos(\theta_x) * \cos(\theta_y) * (F_1 + F_2 + F_3 + F_4)$$

$$\theta_{\text{XY}} = -atan2(F_{\text{prop}_\text{x}}, F_{\text{prop}_\text{y}})$$

$$XY_{\text{2D}} = sqrt({F_{\text{prop}_\text{x}}}^2 + {F_{\text{prop}_\text{y}}}^2)$$

$$F_{\text{prop}_\text{x}} = XY_{\text{2D}} * \sin(\theta_z + -\theta_{\text{XY}})$$

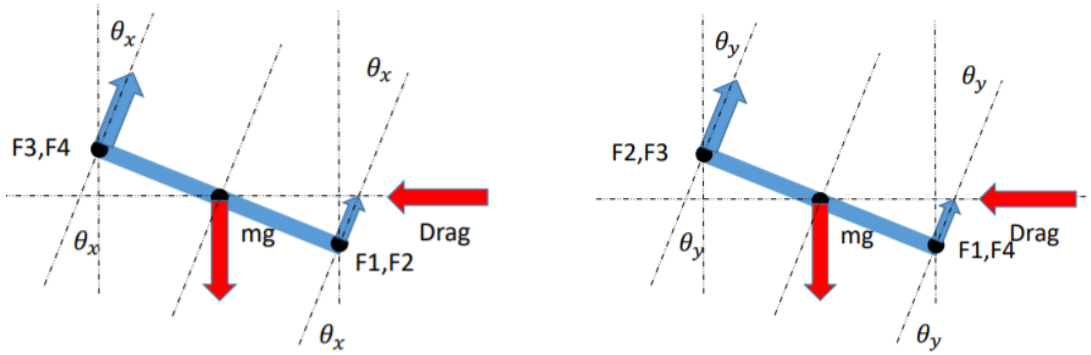
$$F_{\text{prop}_\text{y}} = XY_{\text{2D}} * \cos(\theta_z + -\theta_{\text{XY}})$$

$$Force_x = F_{\text{prop}_\text{x}} + -Drag_x$$

$$Force_y = F_{\text{prop}_\text{y}} + -Drag_y$$

$$Force_z = F_{\text{prop}_\text{z}} - m * g + -Drag_z$$

Here F_{prop_i} is the thrust created by the ith propeller, $Force_i$ is the net linear force on the drone and $Drag_i$ is the net drag force by environment on the drone in the ith axis. For better understanding, see 3.6 and 3.7.



(a) Pitch View Angle (b) Roll View Angle

Fig. 3.6: Pitch and Roll Rotation

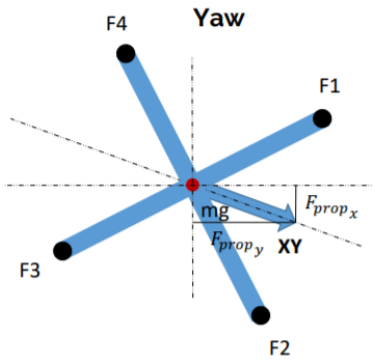


Fig. 3.7: Yaw View Angle

3.3 Controller

Fig 3.8 represent the PID controller used for controlling the altitude, roll, pitch and yaw motions of the drone. By tuning the Proportional, Integral and Derivative parts of the PID controller, faster and flat response can be achieved depending upon the requirements.

3.4 Visualization

Fig 3.9 represent the visualization subsystem to display the the results of simulation. The angular acceleration and linear accelerations are integrated to get linear and angular velocities and positions and are shown as scope blocks for plotting the simulation results.

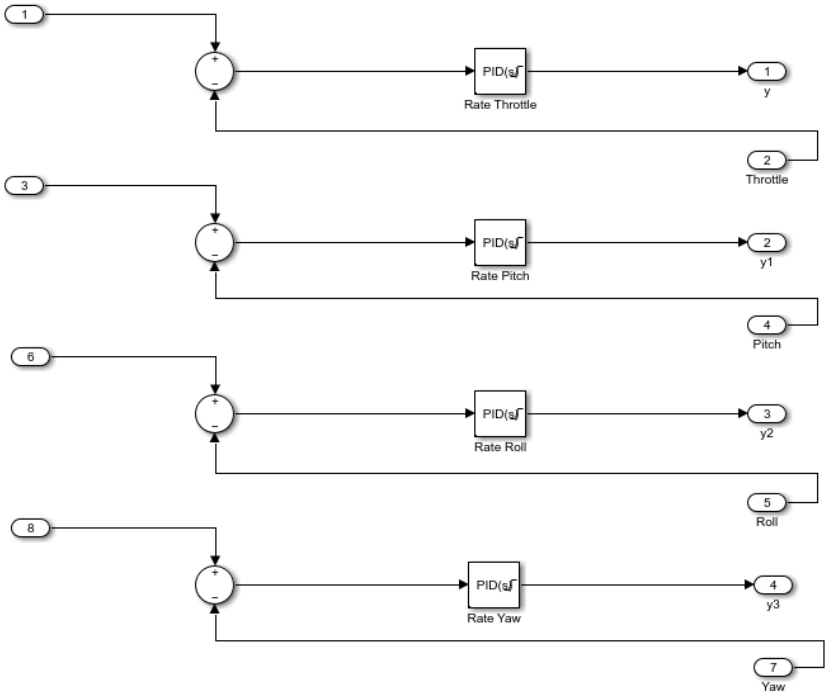


Fig. 3.8: PID Controller

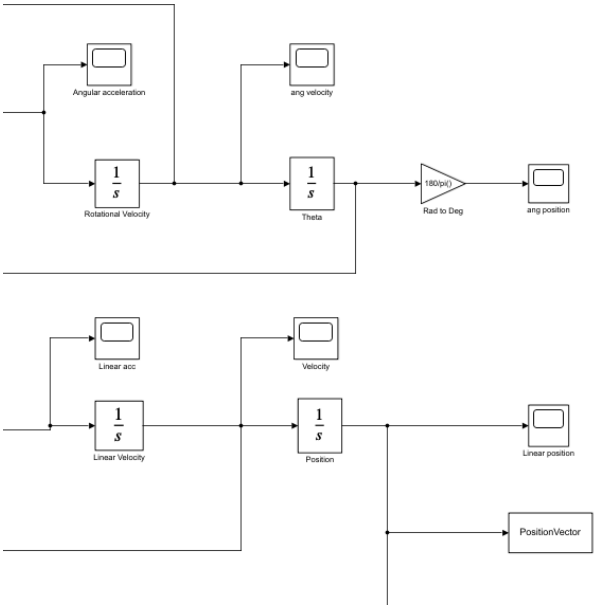


Fig. 3.9: Visualization

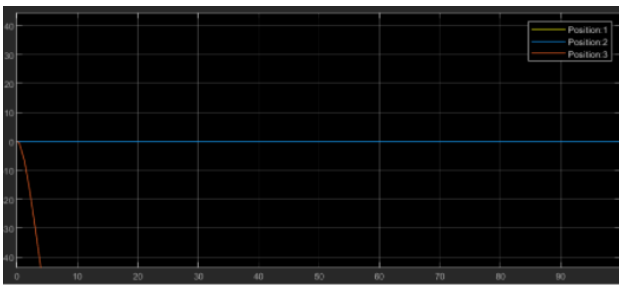
Chapter 4

Simulation

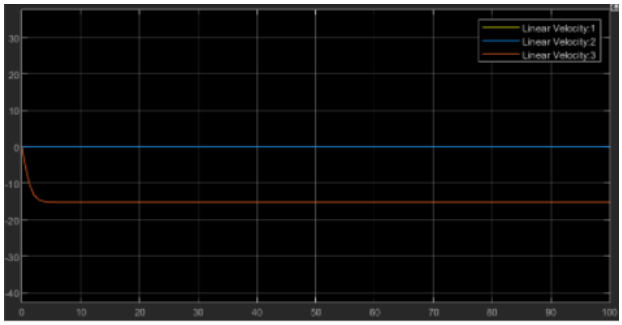
In this chapter, we shall look at the simulation result of the Drone controller cyber physical system. First, we see the simulation when drone is at hover state, then when the drone moves without using PID controller and at last when the drone uses PID controller. We simulate the result for 20 - 100 seconds depending upon the requirements.

4.1 Drone at Hover State

Fig 4.1 shows the falling drone when no thrust, pitch, roll and yaw is applied. Due to air drag, the drone will eventually reach terminal velocity after sometime.



(a) Linear Position vs Time

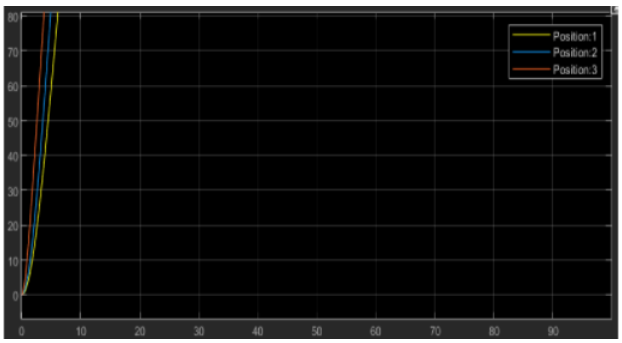


(b) Linear Velocity vs Time

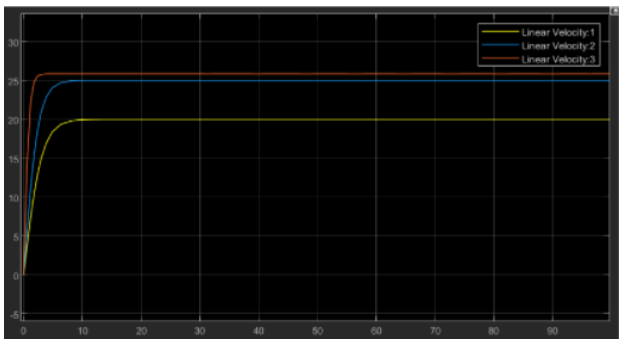
Fig. 4.1: Falling Drone at Hover State

On adding wind gusts of 20 m/s in +x, 25 m/s in +y and 30 m/s in +z direction, leads the

drone to move in +x, +y, +z directions respectively as seen in Fig 4.2.



(a) Linear Position vs Time



(b) Linear Velocity vs Time

Fig. 4.2: Drone at Hover with Wind Disturbance

4.2 High Input Values

Fig **??** shows what happens when the input given to drone is more than 100%. At too much throttle, equal to 200% here, drone is falling and is showing unusual behaviour because maximum throttle is not possible to be achieved.

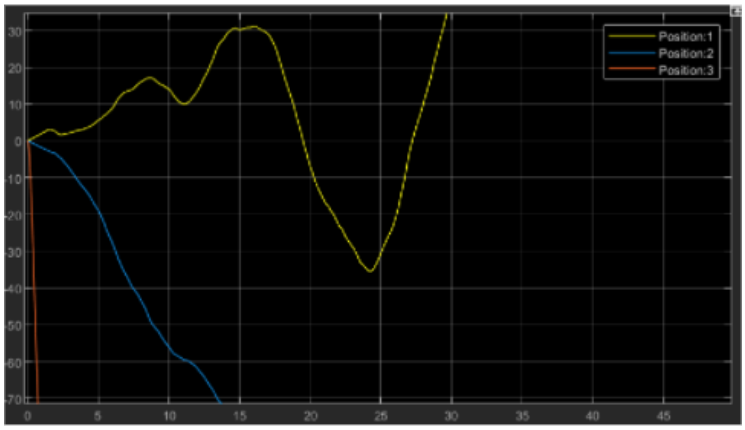
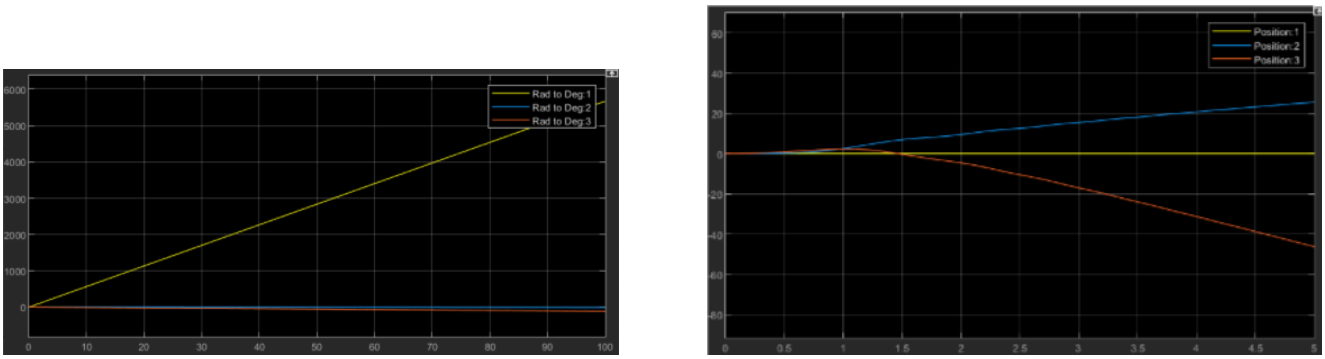


Fig. 4.3: High Throttle

4.3 Pitch with Throttle

Fig 4.4 and 4.5 shows the angular and linear position of drone when Throttle = 50 and Pitch = 1. Since, drone rotates about x axis, so it will not move in x direction. After 180°

rotation about x axis, drone will move in -z direction and +x direction and so on.



(a) Angular Position vs Time

(b) Linear Position vs Time

Fig. 4.4: Pitch with Throttle

In Fig 4.5, after 180° rotation about x axis at 0.75 seconds, drone will decrease it's velocity in z axis, then after 360° 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.

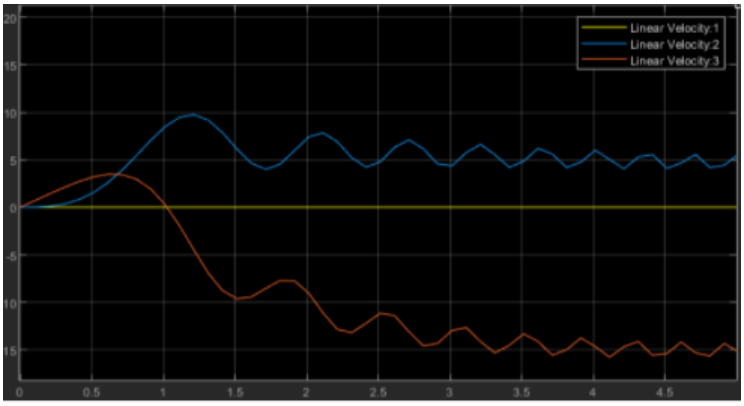


Fig. 4.5: Linear Velocity vs Time

4.4 Roll with Throttle

Fig 4.6 shows the angular and linear position of drone when Throttle = 50, Roll = 0.5. When rotating about y axis, drone will first move in +z direction then after each 180° interval, direction is reversed. Same is followed for x axis also.

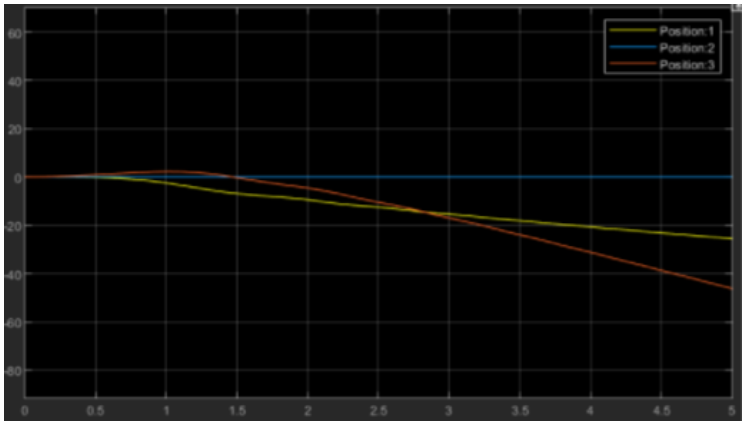


Fig. 4.6: Roll with Throttle

4.5 3D Visualization

Fig 4.7 shows the 3 dimensional view of the drone when Throttle = 50, Pitch = 0.035, Roll = 0 and Yaw = 0. It is plotted by using the below equation in MATLAB.

$$plot3(positionVector_x, positionVector_y, positionVector_z, -r)$$

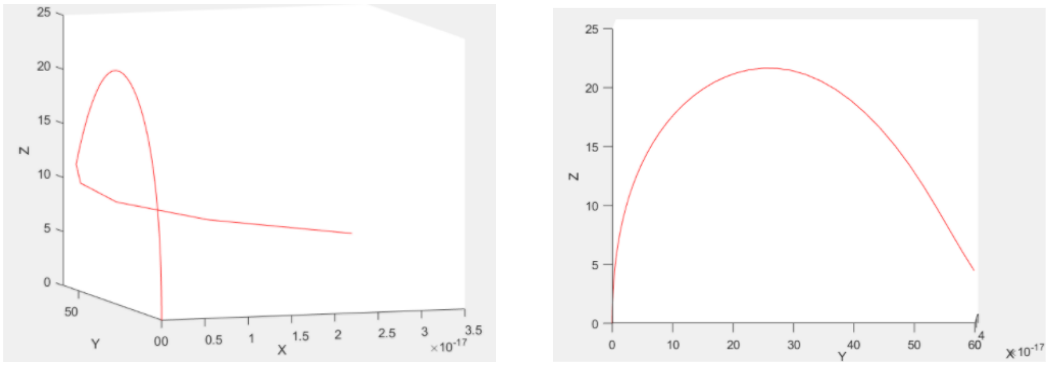


Fig. 4.7: 3D Visualization

4.6 Altitude Control

Fig 4.8 shows the linear position of drone when Throttle = 70 and PID controller is used with $T_P = 5$ (proportional gain), $T_I = 1$ (integral gain) and $T_D = 0$ (derivative gain). Fig

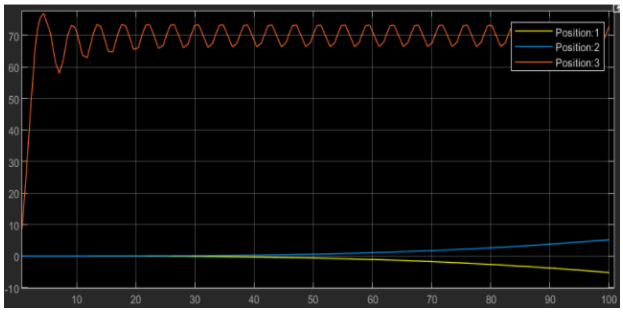


Fig. 4.8: Altitude Control without PID Tuning

4.9 shows the response curve when Throttle = 70 and $T_P = 13.2$, $T_I = 0.8918$ and $T_D = 34.38$. We can see here that when Throttle is controlled by PID controller, it acts as an altitude controller. Fig 4.10 has $T_P = 9.03$, $T_I = 0.5268$ and $T_D = 10$. It shows that faster

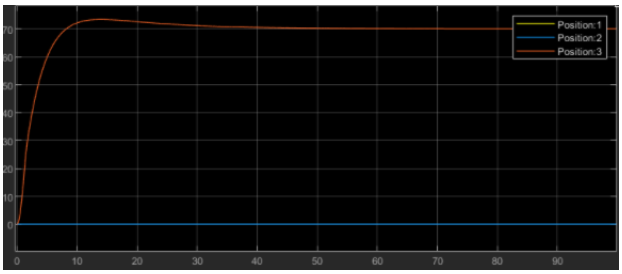


Fig. 4.9: Altitude Control with PID Tuning

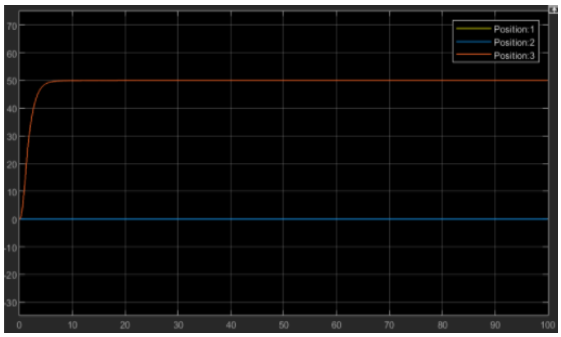
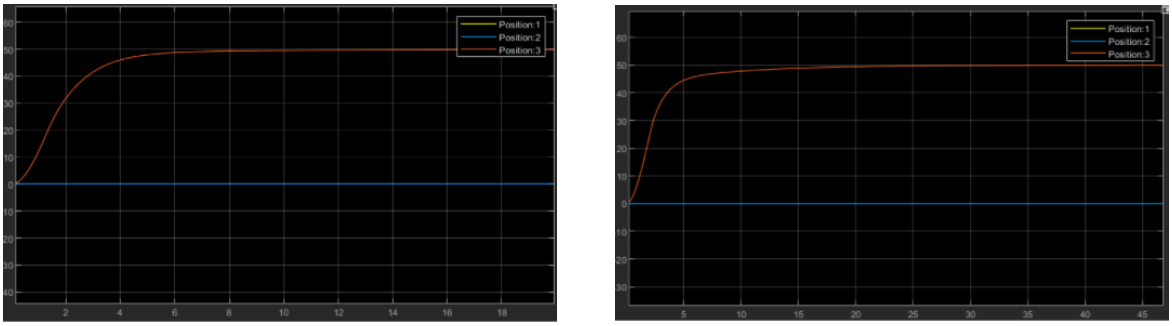


Fig. 4.10: Altitude Control with Faster and Flat Response

and flat response can be obtained by increasing integral gain and decreasing derivation gain after tuning the PID controller.

4.7 Altitude PID Controller and Air Gusts

Fig 4.11 shows the plot of linear position with time along with wind disturbance in +z direction. It shows that when wind velocity is more in +z direction, drone will take less time to reach altitude. First case took around 8 seconds to reach while second case took 15 seconds.

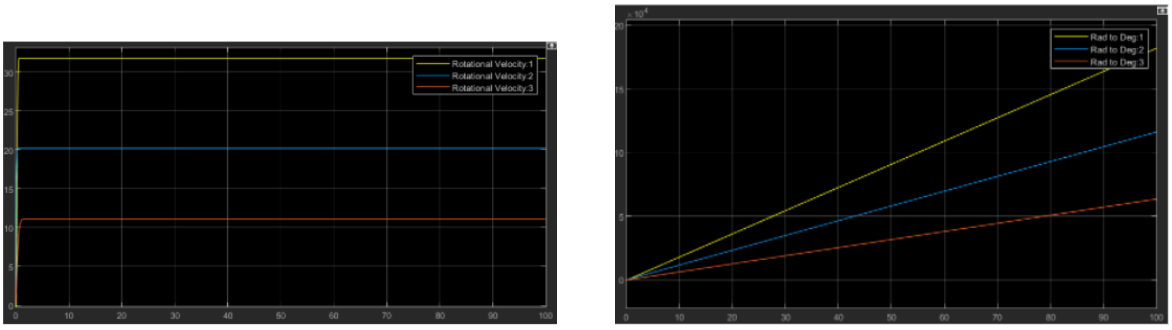


(a) Wind disturbance: -10 m/s (b) Wind disturbance: -20 m/s

Fig. 4.11: Altitude PID Controller and Air Gusts

4.8 Pitch, Roll and Yaw Controller

Fig 4.12 shows the responses of angular position and velocity with Throttle = 50, Pitch = 32, Roll = 20, Yaw = 12 along with PID controller tuning.



(a) Angular Velocity vs Time (b) Angular Position vs Time

Fig. 4.12: Pitch, Roll and Yaw

Chapter 5

Verification

In this chapter, we verify the drone controller cyber physical system using KeYmaera X tool.

5.1 KeYmaera X verification tool

KeYmaera X is a tool for verifying hybrid automata. Hybrid automata are models combining finite state machines and differential equations. Is is used for hybrid systems that combines deductive, real algebraic, and computer algebraic prover technologies. It is an automated and interactive theorem prover for a natural specification and verification logic for hybrid systems. KeYmaera supports differential dynamic logic(dL), which is a first-order dynamic logic for hybrid programs. For automation, KeYmaera implements a free-variable sequent calculus and automatic proof strategies that decompose the hybrid system specification symbolically.

5.2 Hybrid Automata

As KeYmaera X tool is used to verify hybrid automata, we took some assumptions to simplify our model as shown in Fig 5.1. Since it doesn't supports non linear functions like

Sin and Cos, we transformed those equations using some internal variables like a, b, c, d etc.

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 */
```

Fig. 5.1: Assumptions

Fig 5.2 shows all the internal state variables, state variables and constant terms used while performing verification.

5.3 Verification of Drone Controller

Fig 5.3 shows all the input values and continuous, discrete part of the dynamics are separated from each other.

5.3.1 Verification of Properties

At the end of Fig 5.4, the invariant is used to verify the property whether the condition is true or not after sometime. We are checking here whether the altitude of drone is within

```
/* 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.
```

Fig. 5.2: Variables and Constant Terms

certain range or not (treating it as an altitude controller) after sometime. Similarly we can perform verification for pitch, roll and yaw angles also.

```
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;

```

Fig. 5.3: Equations and Modelling: 1

```
            /* 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.
```

Fig. 5.4: Equations and Modelling: 2

Chapter 6

Conclusions

We explored the Drone or UAV controller as a cyber physical system. We modeled and simulate the system in MATLAB and Simulink and observe the output under different values of Throttle, Pitch, Roll and Yaw and studied them.

We conclude from the simulation results that the altitude and angular velocities of the drone can be controlled using PID controller even though wind gusts and disturbances are present in the environment. Also the drone is having limited power which makes it more realistic to be used in real life.

We explored the KeYmaera X tool, which is used to verify the hybrid systems using automatic proof strategies that decompose the hybrid system specification symbolically. We create an overall code system representing the drone dynamics and tried to verify properties like altitude and rotation control.

References

- For basics of MATLAB and Simulink:

  https://www.mathworks.com/learn/tutorials/matlab-onramp.html

  https://www.mathworks.com/learn/tutorials/simulink-onramp.html

- Download MATLAB and Simulink:

  https://in.mathworks.com/downloads/web_downloads

- For modeling and simulation of Temperature Controller:

  https://in.mathworks.com/videos/drone-simulation-and-control-part-1-setting-up-the-control-problem-1539323440930.html

- Data Values for Plotting Equations:

  https://www.apcprop.com/technical-information/performance-data/

  https://www.apcprop.com/files/PER3_8x4.dat

- Download KeYmaera X:

  https://keymaerax.org/keymaerax.jar

- For Verification using KeYmaera X:

  https://www.cs.cmu.edu/ smitsch/pdf/KeYmaera-tutorial.pdf

- My Overall Work can be seen here:

  https://github.com/SAMAYV/CS591_Simulation_Verification