LEUPHANA
UNIVERSITÄT LÜNEBURG

Field of study: Master of Science Management and Data Science

Project on the module Modelling and Control of Dynamical Systems using Linear and Nonlinear Differential Equations

---

A Simulation of Modelling And Control of 2D UAV to reach a Target

---

Submitted by
Sthitadhee Panthadas (3044093)

Lüneburg, 20.03.2023

Examiner: Prof. Dr. Ing. Paolo Mercorelli

Statuatory declaration

I hereby certify that this project has been written independently and that no sources or aids other than those indicated have been used, and that all passages in the thesis that have been taken verbatim or in spirit from other sources have been marked as such.

Lüneburg, 20.03.2023

Sthitadhee Panthadas

Abstract The project implements a game on target reaching using UAV as an game object that tries to reach a target balloon. It is an imitation of many similar games but with the focus of depicting the modelling side using the differential equations namely the equations of motion and the controlling side using PID (Proportional, Integral and Derivative). The outlook of the project for modelling and control is show how to approach simulating equations of motion model using GUI (graphical user interface) with a special emphasis of working with C++ and python. The main aim was to revise concepts of control and modelling and programming concepts in C++ and python and finally to see how they can be used for simulation and analysis.

The project reinvents similar implementations using C++ and Python and tries to connect different technologies such as Matlab for inputs for tuning parameters for PID. The final output is a simulation or an autonomous game. VScode and Matlab was the go to softwares for the study and running of the developement process.

In the results, it can be seen how simple differential equations can be used to create simulations and game concepts. The project also can be used as an effective tool for experimentation by adding new concepts such as my other project in optimization techniques course.

Acknowledgement I would like to acknowledge my brother's (Panthadas, Himadri) support for explaining to me the concepts of PID controllers and helping me with Matlab/Simulink and making a correction on one of the diagrams. Finally, I would acknowledge professor Paolo Mercorelli for always providing us the necessary guidance throughout this semester.

Contents

1 Introduction

In the Introduction I will briefly explain the context of modeling and control of dynamic systems using linear and non-linear context and then trace the itinerary of the project planning and reference some of them from literature.

Modeling is the process of creating a simplified but cohesive representation of a real world system [7]. A model captures the characteristic features of a system and the system's essential behaviors as well as constraints. This enables the use of the model to predict future behavior, describe past behavior or test hypotheses. On the other hand, controlling refers to the process of manipulating the model or system to a desired outcome based on the system's inputs and some manipulative actions. Some examples of models can be probabilistic models (bayesian models), markov models, game theory models, and so on. However, here we are focusing on differential equation models having a combination of physical, mathematical and computational elements.

Throughout the project planning many differential equations and models were researched for the project such as lotka-volterra [5] [8] [9], Maxwell's equation, Duffing equation, Dirac Equation, Van der pol system, Hamilton's equation, Navier Stoke's equation, Black-Sholes equation, Diffusion equation, Schrodinger's equation and many more. Furthermore, a recently established research of neural ODE [3] and its applications were also researched for the project. Finally, based on the time constraint, the adequate ease of implementation, learning curve and maximum output the current project was chosen for implementation at the end.

In the case of control, only Proportional-Integral-Derivative (PID) controllers [6] [2] (PD/PI etc as well) was researched for the realization of the project.

The project is a rebuilding of a youtube video [4] with a focus in modeling and control which implements controlling of UAV using a control mechanism to reach a target using rigid body mechanics of the UAV and equations of motions. Equations of motions act here as the differential equation model for the UAV and the target is represented by a balloon and the control technique used for the purpose of reaching the

goal of the project is the PID controller. Some image resources are also obtained from here.

The rest of the structure of the report is the following. `Methodology` will be a detailing of the grounding knowledge or theoretical design of the whole process implemented here in the project. This will include all the necessary theoretical concepts and mathematical formulations in a short concise format. Then we will have 3 sections for the `Results` section namely, `Code` and `Simulation` and `Graphs`. This will contain the main body of the work and the results obtained for the project overall. Finally, at the end we will have a `Discussion` section which will delineate the positives, limitations and future scopes of the project. There will also be `supplementary materials`, `list of figures` and `list of acronyms`.

2 Methodology

2.1 Theoretical Design

The task of implementing a simulation of an UAV reaching a specified target can be implemented with the following theories - rigid body diagram [1] and equations of an UAV or simply UAV dynamics, equations of motion and in this case PID controller [6]. For the physics of the UAV the following assumptions are made: movement rightwards and downwards is positive direction and movement leftwards and upwards is negative. If they are reversed then all the signs for the vectors forces, velocities etc will be reversed. The concept of Newton's laws of motion is also required here.

The following will be the concise methodology that implements the logic of the project.

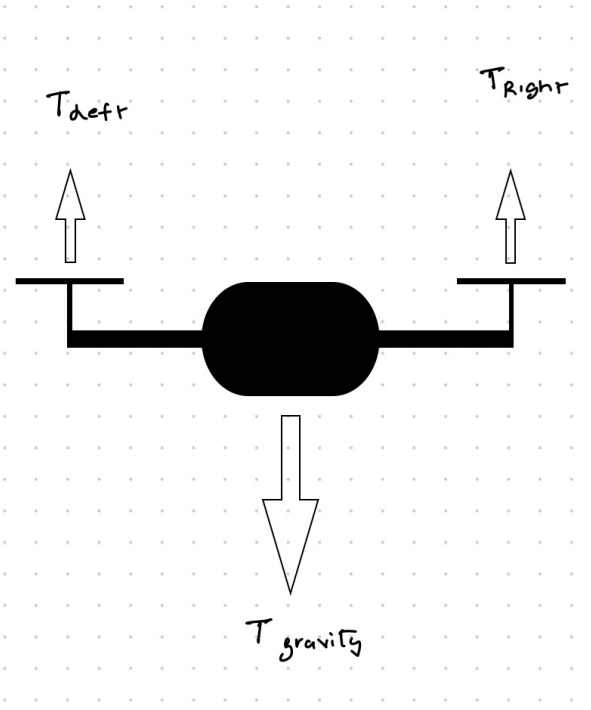

Figure 2.1 Rigid Body Diagram of UAV

Rigid Body Diagram The rigid body diagram for a 2D UAV is used. In a 2D UAV we can see 3 active forces at play at one given point or in game terms per frame. The first two are the left and right thrust for each propeller acting upwards and the force of gravity acting downwards. Let the mass of the UAV be m.

Now that the main forces acting on the rigid body are realized, it is important to look at the three specific dynamics of the system.

As per Figure 2.1 we can see the system in equilibrium and the UAV is not moving in the y axis or in the x axis. Here, the upward forces balance the weight of the UAV. There are no resultant forces in the y or x direction.

$$\Delta T = 0$$

$$T_{left} + T_{right} - T_{gravity} = 0$$

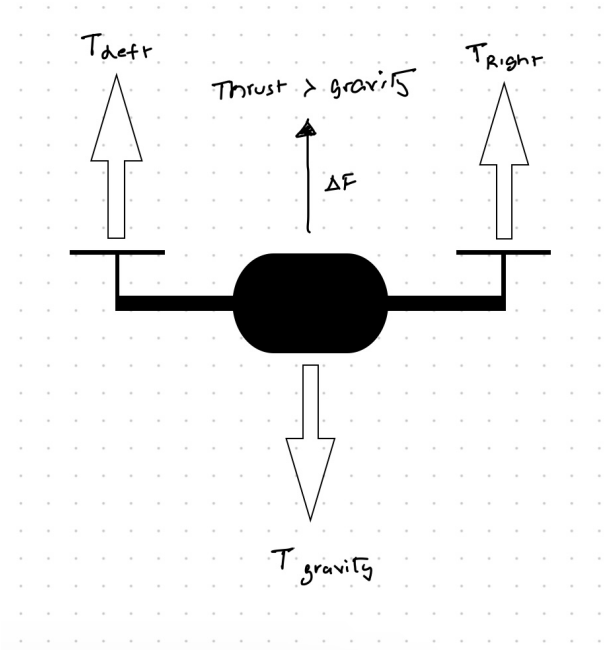

Figure 2.2 Rigid Body Diagram of UAV going up.

As per Figure 2.2, we can see the system is in motion in the y axis and the resultant force if upwards will move in the upward direction or downwards will move in the downward direction. If the resultant force is upwards, the *thrust* > *weight* and if it is downwards *thrust* < *weight*.

$$\Delta T = ma$$

$$a = \frac{T_{left} + T_{right} - T_{gravity}}{m}$$

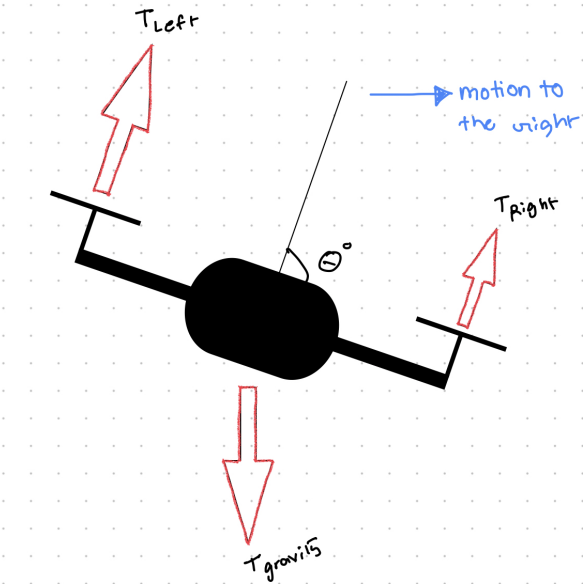

Figure 2.3 Rigid Body Diagram of UAV going left. T_{Left} being greater in magnitude than T_{Right} leading the UAV to tilt right.

As per Figure 2.3, we can see the system in motion in the x axis. Here, it is important to see that the θ is not 0. When the θ is greater than 0 the dynamics of the UAV become the following. The signs of the vectors will change if opposite directions are thought of as positive as mentioned before.

$$\ddot{s}_x = -\frac{T_{Left} \cdot \sin\theta + T_{Right} \cdot \sin\theta}{m} \tag{2.1}$$

$$\ddot{s}_y = -\frac{T_{Left} \cdot \cos\theta + T_{Right} \cdot \cos\theta - T_{gravity}}{m} \tag{2.2}$$

$$\ddot{\theta} = -\frac{-T_{right} + T_{left}}{m} \cdot l \tag{2.3}$$

where $\ddot{s}_x$ is the acceleration in the x direction and $\ddot{s}_y$ in the y direction and $\ddot{\theta}$ is the angular acceleration. T_{right} is the thrust right, l is the length of the arm of UAV from the center.

Equations of Motion The following equations of motion or differential equations are taken to calculate the velocity and position of the UAV.

$$v = u + at \tag{2.4}$$

$$s = vt \tag{2.5}$$

PID controller Proportional, integral and differential control is a closed control and a feedback system used for numerous applications. In the context of the project, the PID controller ensures speed, response and stability for a UAV game object in reaching a target balloon. PID controllers generally take the difference between an input and an output result and then try to quickly reduce that error to the required amount. Here, the PID controller takes the difference between the position of UAV and target in the x and y axis as the error measure. Since it does not make sense in real life situations or sometimes is not possible to nullify the difference or the error in 1 step, the nullification process is done over a period of time.

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \tag{2.6}$$

Where $u(t)$ is the control signal, $e(t)$ is the error which is at each interval over the full period of time. *Kp*, *Ki* and *Kd* are coefficients of proportional error, integral error and differential error respectively. The proportional error is actually proportional to the error itself. The lower the value, the more slowly the nullification happens, the bigger the constant, the higher the instability of the system. Integral error can be thought of as the amount of accumulated error over time. Large *Ki* results in oscillations. *Kd* is the constant that determines how quickly the system settles from overshoot and is sensitive to noise in the system. It is generally a small value.

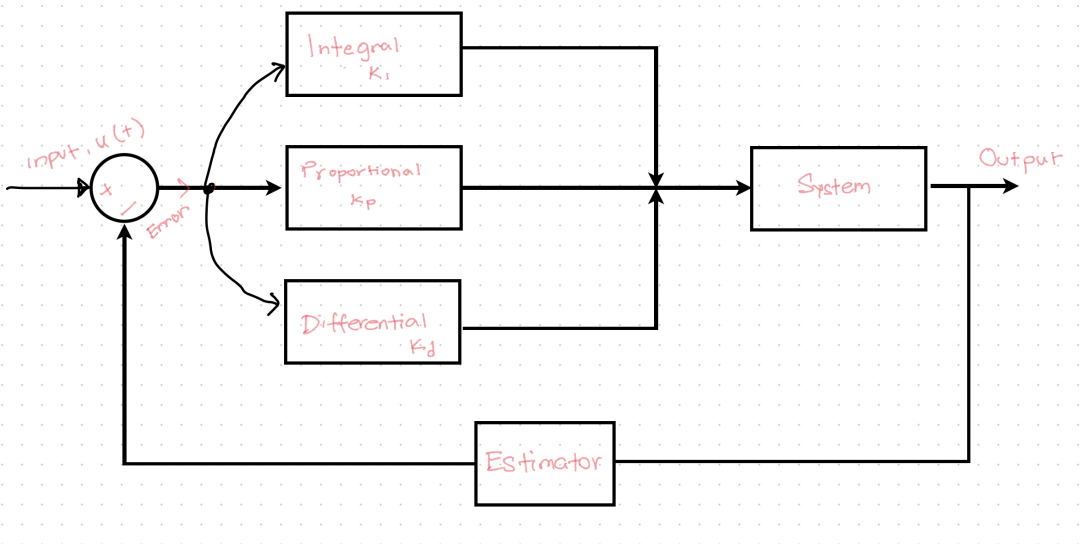

Figure 2.4 Pid Controller Overview

2.2 Code

In this section, we will talk about the code and code structure followed. Here the description will be somewhat of a general idea of how the code works and how it is structured and aims to abstract the higher technical details of the code.

Pseudocode Below the pseudocode is also representative of the above mentioned principle/s followed to describe the code structure.

```
START

Initialise the variables
Create the textures
Generate r random positions for target
Initialise different clocks for animation
Animate the cloud
Instantiate the PID player
Loop as long as window open:
    Clear window
    Handle ending of game and close window event
    Handle cloud, target and flying object animation
    Clear window
    Loop through players
        Set initial y acceleration, x acceleration and angular acceleration
        Calculate next x and y thrust of UAV based on PID action
        Calculate the x, y and angular acceleration
        Calculate updated velocities
        Calculate updated position
        Store updated values // for graphing
    Draw
    Calculate new fps to be updated at next loop

END
```

File Structure Figure 2.5 shows you the file structure of the project. Although the code is adequately commented, a brief description of the main files and folders are provided below.

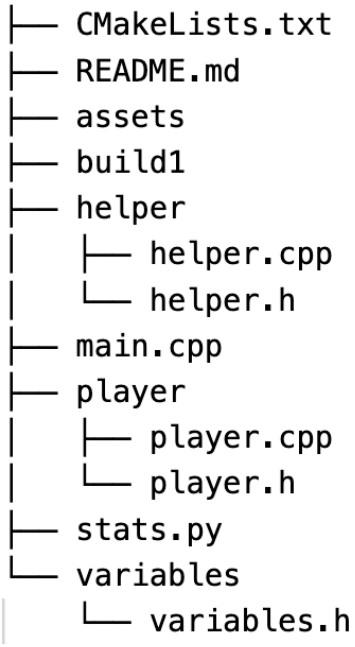

Figure 2.5 File Structure of the Project.

`Main.cpp` that gets run first by the compiler.

`assets/` folder contains the assets required for this project.

`helper/` folder that contains the declarations and implementations of helper functions.

`player/` folder that contains the declarations and implementations of classes for each player and their properties and methods. Here it contains the class of the PID player and also contains the class required for implementation for PID control systems. PID could have been abstracted away to its own folder but it is avoided for now.

`CMakeLists.txt` file that contains the setup for cmake for building and running the code.

`stats.py` file that contains the python code that graphs distance vs time, velocity vs time, acceleration vs time for x and y dimensions.

3 Results

3.1 Project Results

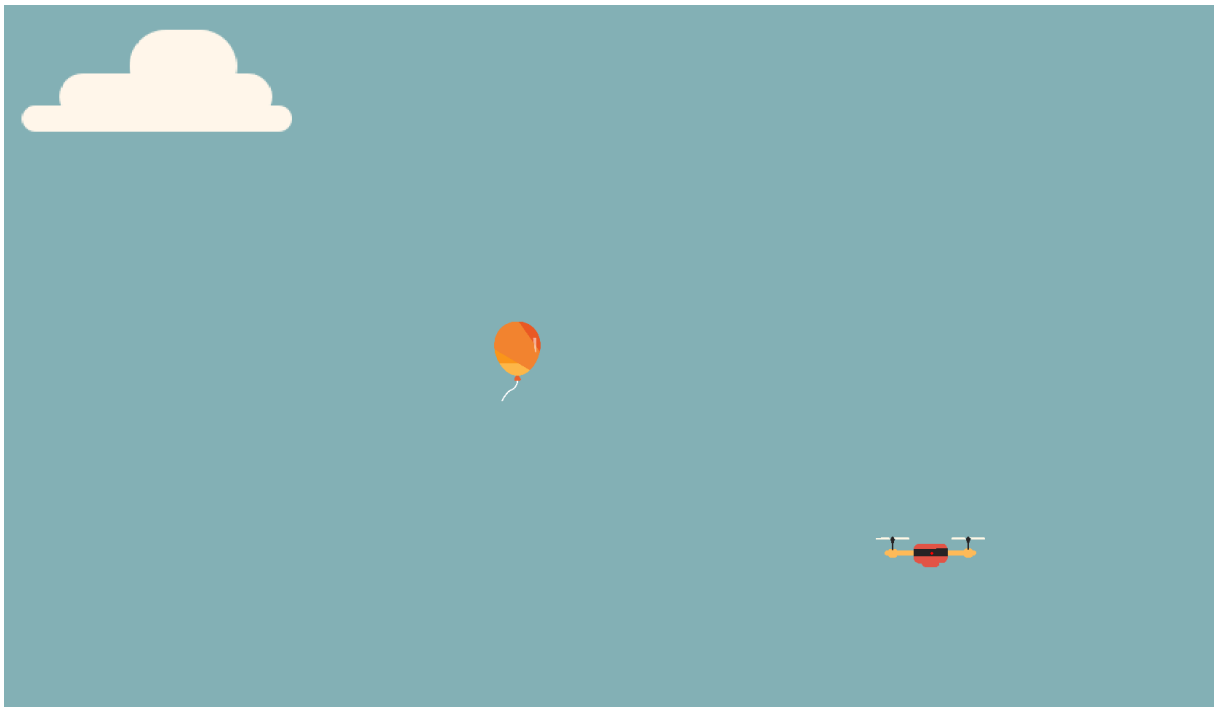

Figure 3.1 An image of the simulation.

Simulation The simulation starts with the UAV in the center of the window and the target balloon created at a random position within a boundary of the window. The whole window is not the boundary because this will cause the UAV to move out of the screen due to the inertia of the UAV after reaching a target that might be on one corner side of the window. To avoid this case, the targets are generated having an invisible padding.

When the application is run, the UAV uses the PID control logic to calculate the left and right thrusts required based on its position and the position of the target. This discrepancy acts as the error input for the PID controller.

The updated thrust will be used to calculate the new acceleration using the rigid body diagram equations and then the new velocity and the new position. This is repeated until the UAV reaches its target and the position of the target updates when the target is within the UAV's range. The process is repeated n times as required and then the UAV stabilizes to rest state eventually.

Details about running the project can be seen in the README.md file of the project.

Graphs As the UAV reaches the target, there is an effort to store all the update values made by using PID controller. The x axis marks the total period from rest till the UAV reaches the target. The values are then used to draw graphs as shown below in 3.2.

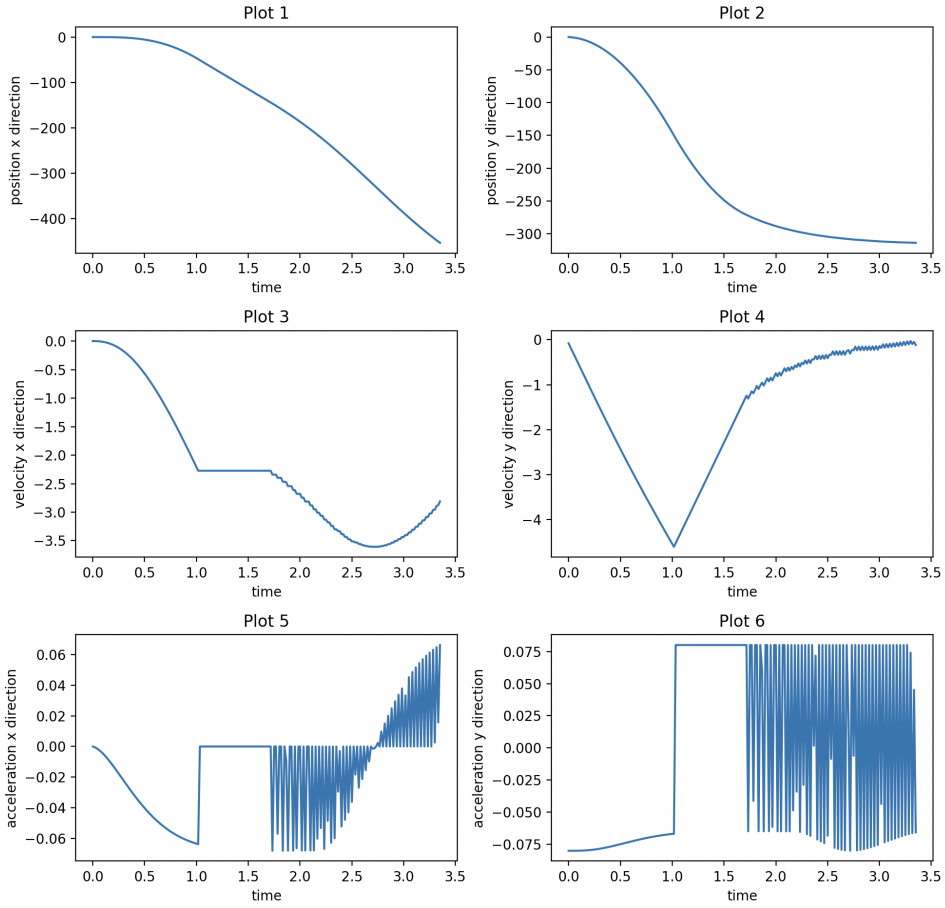

Figure 3.2 Graphs corresponding to Figure 3.1 till the point of reaching Target.

This has been shown here for 2 reasons. First to imitate the lectures of the classes and its intended goals. Another reason could be an ad hoc method to quickly compare the acceleration, velocity and position updates of the simulated UAV animation to that of a real UAV scenario. This can be used to quickly understand how well the

simulation can be applicable for a real case scenario. Depending on the graphs we might want to change the heuristics or hyper parameters of the PID controller. Of course, the values of thrust and gravity need to be properly scaled to real life cases as well.

Figure 3.2 is the resulting graph of the setting of 3.1. It can be seen that the going upwards and leftwards is resulting in negative values in the graph. This means downwards and rightwards is the positive direction. We can see in the first two plots the position of the UAV reaches the target in 3.5 seconds following the following dynamics shown by the curve. The corresponding changes of the velocity and acceleration for both axis can be seen. We can see as the UAV moves closer to the target, the changes is becoming more responsive and erratic. To see the stabilising of the UAV position we have to record a bit further than 4 seconds. This can be seen in the Figure 3.3 below. It can be seen the UAV stabilises at the desired position after some time.

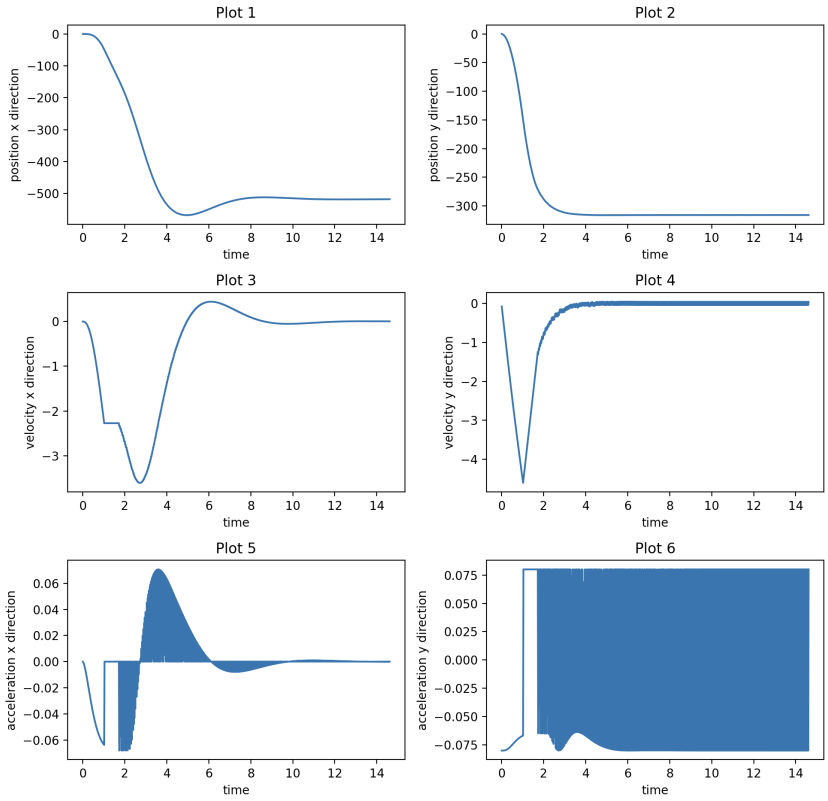

Figure 3.3 Graph Statistics till stabilisation.

Matlab Simulink There was also an effort to create a simulink version of the problem simulated in the application. The simulink file will also be provided. Below is a Figure 3.4 showing the design.

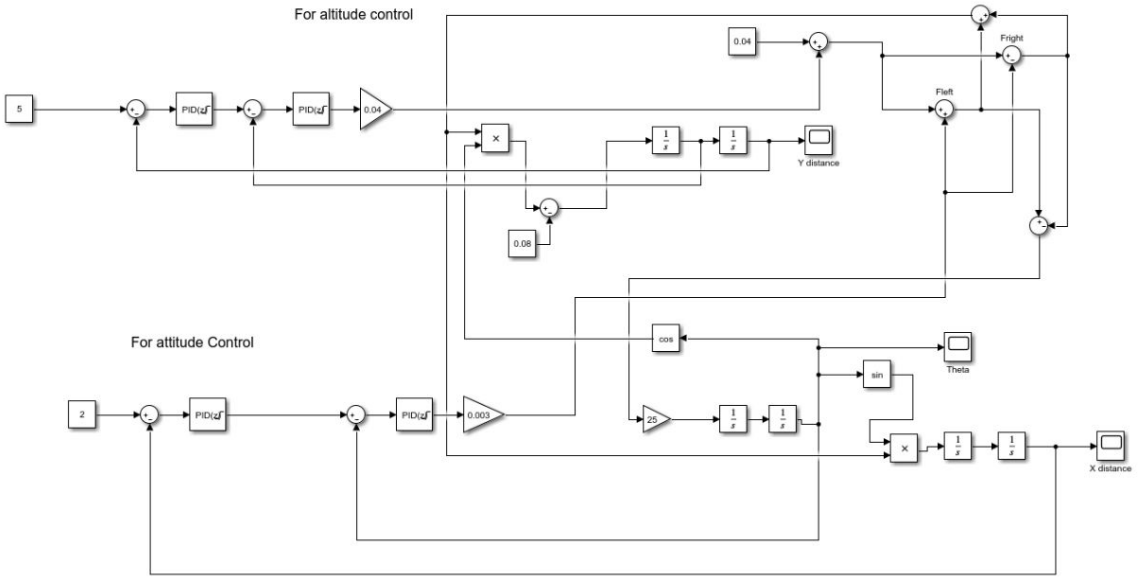

Figure 3.4 Matlab Simulink Model.

The desired set point of y is given at 5 and for x it is 2 in this model. The current position (both x and y distance), velocity and angle to the vertical are continuously monitored and the errors are calculated. For the altitude control using the error in Y distance and the PID block we generate the required velocity and its error also calculated. It is passed through a second PID control whose output creates the required changes in the forward thrust.

For the attitude control using the error in X distance and the PID block we estimate the required angle or orientation of the drone. The error for the angle is then calculated and with another PID block we create changes in the difference in between the thrust for its angular motion. The scopes are named as Y distance, X distance and theta and shows the current Y position, X position and the angular orientation of the drone respectively. They are given below in Figure 3.5.

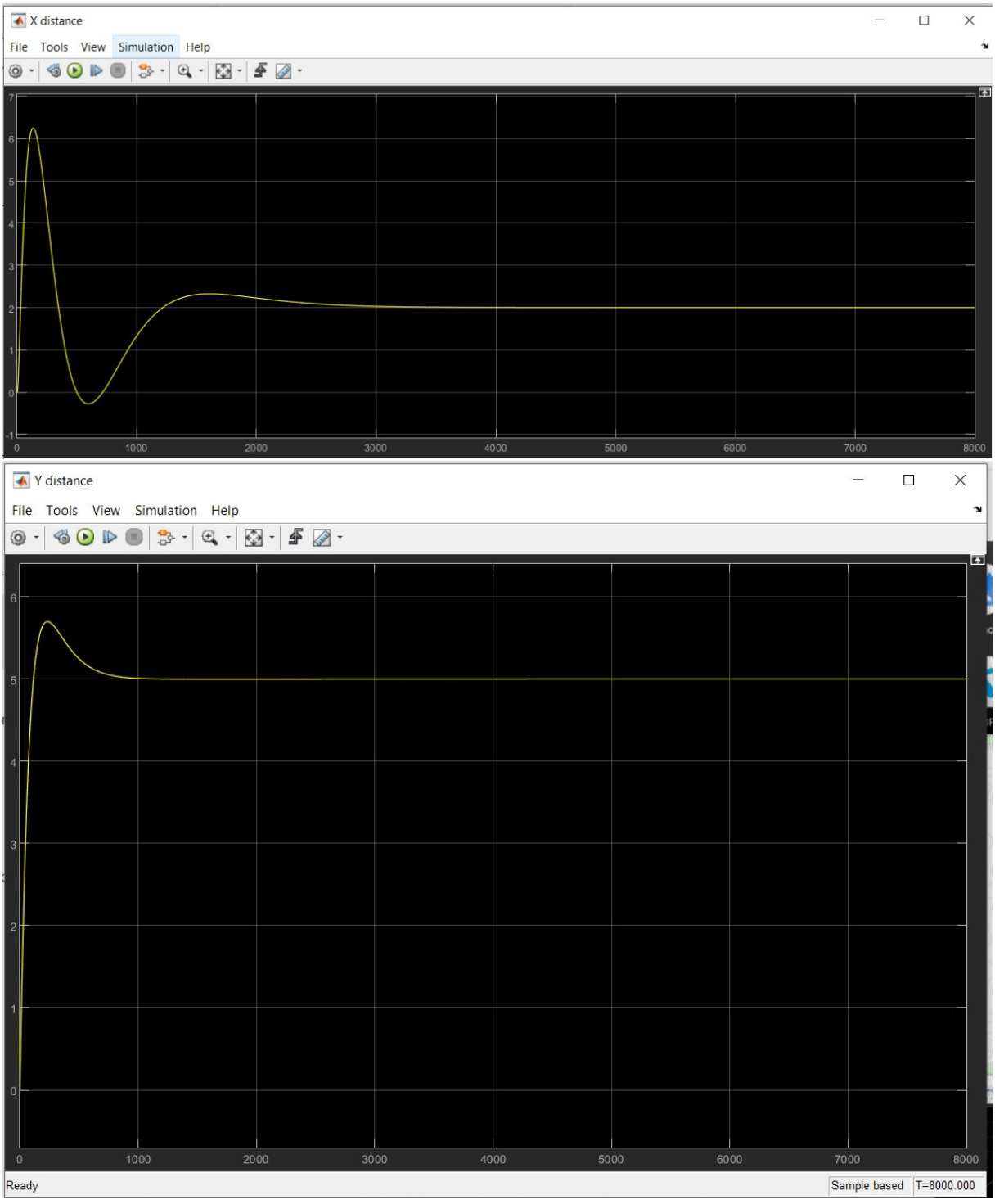

Figure 3.5 Matlab Simulink Graph of scopes.

4 Discussion

Overall, the project implements a simple game like GUI based project where 2D UAV can reach a target based on laws of physics and a control mechanism called PID control.

The project was a huge learning experience in multidimensional ways. The project enabled the application of C+, SFML library, python, matlab/simulink for modeling, Freeform for some of the graphics and latex for report writing.

The process of development of this project led to the following realizations:

- Understanding how and when modeling and controlling can be applied in real life applications
- Understanding how different tools can help to create real valued projects for modeling and control

It also led to some learning and revision:

- Different concepts in differential equations, modeling and control
- Feedback loops
- Simulink and C++

Although the project in itself is complete, it is not a great interactive experience for the user. Due to the limitation of time, further enhancement that would increase the lucrativeness of the project through more features and better animations could not be achieved. Also, code can be further optimized and further abstractions are possible along with code breakdowns. Finally, it would be great if the hyperparameters from simulink could be generated such that they could be used in the application. However, due to short time for research and implementation in this area, the implementation has been avoided in this project.

However, all in all the aim of the course for Modeling and Control has been realized by this project.

5 Supplemental Material

See the electronical attachment "modellingControlUav.zip". This zip contains all the files including the simulink file.

List of Acronyms

GUI	Graphical User Interface
PD	Proportional-Derivative
PI	Proportional-Integral
PID	Proportional-Integral-Derivative
SFML	Simple and Fast Multimedia Library
UAV	Unmanned Aerial Vehicle

List of Figures

List of References

[1] *2D Rotorcopter Mechanics and PID Control with Unity - The Applied Architect*. URL: http://www.theappliedarchitect.com/learning-2d-rotorcopter-mechanics-and-control-with-unity/.

[2] *3.3: PI, PD, and PID Controllers*. URL: https://eng.libretexts.org/Bookshelves/Industrial_and_Systems_Engineering/Book%3A_Introduction_to_Control_Systems_(Iqbal)/03%3A_Feedback_Control_System_Models/3.3%3A_PI%2C_PD%2C_and_PID_Controllers.

[3] Ricky T. Q. Chen et al. *Neural Ordinary Differential Equations*. 2018. DOI: 10.48550/ARXIV.1806.07366. URL: https://arxiv.org/abs/1806.07366.

[4] *Controlling Drones with AI (Python Reinforcement Learning Quadcopter)*. URL: https://www.youtube.com/watch?v=J1hv0MJghag.

[5] P.H. Kloppers and J.C. Greeff. 'Lotka–Volterra model parameter estimation using experiential data'. In: *Applied Mathematics and Computation* 224 (Nov. 2013), pp. 817–825. DOI: 10.1016/j.amc.2013.08.093. URL: https://doi.org/10.1016/j.amc.2013.08.093.

[6] Kiran Kumar Lekkala and Vinay Kumar Mittal. 'PID controlled 2D precision robot'. In: *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*. IEEE, July 2014. DOI: 10.1109/iccicct.2014.6993133. URL: https://doi.org/10.1109/iccicct.2014.6993133.

[7] *Living Textbook: Home: By ITC, University of Twente*. URL: https://ltb.itc.utwente.nl/page/491/concept/79704.

[8] *Lotka–Volterra equations*. URL: https://en.wikipedia.org/wiki/Lotka%5C%E2%5C%80%5C%93Volterra_equations.

[9] Dzhakhongir Normatov and Paolo Mercorelli. 'Parameters Estimation of a Lotka-Volterra Model in an Application for Market Graphics Processing Units'. In: *Annals of Computer Science and Information Systems*. IEEE, Sept. 2022. DOI: 10.15439/2022f61. URL: https://doi.org/10.15439/2022f61.