
**TRIBHUVAN UNIVERSITY**

**INSTITUTE OF ENGINEERING**

**PULCHOWK CAMPUS**

**A**

**PROJECT REPORT**

**ON**

# GUIDANCE, NAVIGATION AND CONTROL OF A VTOL VEHICLE TO MAKE IT FOLLOW A PREDETERMINED TRAJECTORY

**SUBMITTED BY:**

SAKAR PATHAK (PUL075BEI030)

SAMUNDRA ACHARYA(075BEI032)

SHREEJAN SINGH SILWAL(075BEI035)

SWAYAM DHAKAL(075BEI046)

**SUBMITTED TO:**

DEPARTMENT OF ELECTRONICS & COMPUTER ENGINEERING

**April 30, 2023**

# Page of Approval

TRIBHUVAN UNIVERSIY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS
DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

The undersigned certifies that they have read and recommended to the Institute of Engineering for acceptance of a project report entitled **GUIDANCE , NAVIGATION AND CONTROL OF A VTOL VEHICLE TO MAKE IT FOLLOW A PREDETERMINED TAJECTORY FOR REUSABILITY** submitted by **Sakar Pathak**, **Samundra Acharya**, **Shreejan Singh Silwal**, **Swayam Dhakal** in partial fulfillment of the requirements for the Bachelor's degree in Electronics & Computer Engineering.

.....
Supervisor
**Nanda Bikram Adhakari**
Assistant Professor
Department of Electronics and Computer Engineering,
Pulchowk Campus, IOE, TU.

.....
Internal examiner
.....
Assistant Professor
Department of Electronics and Computer Engineering,
Pulchowk Campus, IOE, TU.

.....
External examiner

.....
Assistant Professor
Department of Electronics and Computer Engineering,
Pulchowk Campus, IOE, TU.

Date of approval:

# Copyright

# Acknowledgments

We are overwhelmed in all humbleness and gratefulness to acknowledge the contributions of all those who have helped us to put together these ideas. We have been inspired by many people, institutions and situations for undertaking this difficult yet rewarding task.

We would like to express our humble gratitude to all who directly or indirectly helped us to come up with the idea of this project. Our special gratitude to our supervisor **Dr. Nanda Bikram Adhikari** for his valuable guidance and supervision as well as financial support. We are also grateful to the **Department of Electronics & Computer Engineering** for providing the opportunity to take part in this project.

On top of that, we would like to thank whoever is going to help us accomplish our goals by helping us intellectually, motivating us, helping us financially or otherwise. Our gratitude is also reserved for those who have taken keen interest in our project.

# Abstract

In interplanetary missions, the landing of space vehicles is typically accomplished using parachutes. However, this simple method is not without its challenges, as these vehicles are prone to parachute drifts that are difficult to predict, especially on planets with dense atmospheres like Earth. As a result, significant attention has recently been given to the development of active control systems for space vehicles, allowing for precise guidance, navigation, and control over predetermined trajectories and enabling soft and accurate landings on planetary surfaces.

The ability to follow a predetermined path and land softly and precisely using real-time onboard control algorithms would greatly enhance the capabilities of vehicles for interplanetary travel, while also increasing the re-usability of space vehicles. This not only benefits interplanetary travel but also improves space payload delivery systems by reducing costs and increasing efficiency.

To this end, this project aims to implement control algorithms on an Electric Ducted Fan (EDF) powered model of a Vertical Take Off and Landing (VTOL) vehicle, enabling it to follow a fixed trajectory. A small CanSat payload will be attached to the vehicle and deployed at a specific altitude, simulating the tasks required of a full-scale vehicle.

By utilizing these advanced control systems, space vehicles can navigate more accurately and efficiently, reducing the risks and costs associated with interplanetary travel. With a focus on trajectory control and precision landing, this project aims to contribute to the ongoing efforts to enhance space exploration and technology development.

Keywords: *Electric Ducted Fan (EDF), Vertical Take Off and Landing (VTOL), Trajectory*

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| **GNC** | Guidance, Navigation and Control |
| **VTOL** | Vertical Take Off and Land |
| **EDF** | Electric Ducted Fan |
| **ICBM** | Intercontinental Ballistic Missile |
| **TVC** | Thrust Vectoring Control |
| **MPC** | Model Predictive Control |
| **ESC** | Electronic Speed Controller |
| **LQE** | Linear Quadratic Estimator |
| **LQR** | Linear Quadratic Regulator |
| **LQG** | Linear Quadratic Gaussian |
| **PRBS** | Pseudo Random Binary Sequence |
| **PD** | Proportional Derivative |
| **PID** | Proportional Integral Derivative |
| **KP** | Proportional Gain |
| **KI** | Integral Gain |
| **KU** | Critical Gain or Ultimate Gain |
| **TU** | Ultimate Period. |
| **OCP** | Optimal Control Problem |
| **NLP** | Nonlinear Programming Problem |
| **IPOPT** | Interior Point Optimizer |
| **QP** | Quadratic programs |
| **UAVs** | Unmanned aerial vehicles |
| **CPU** | central processing unit |
| **GPS** | Global Positioning System |
| **IMU** | Inertial Measurement Unit |
| **PWM** | Pulse Width Modulation |
| **SCL** | Serial Clock |
| **SDA** | Serial Data |

# 1. Introduction

## 1.1 Background

This project is being undertaken by a team of four undergraduate students who are interested in learning how rockets work. Rockets have been essential in space exploration and scientific research, and the team believes that gaining a deeper understanding of how they function will provide valuable insights into the field of aerospace engineering.

The team's primary objective is to develop GNC algorithms for a VTOL vehicle that replicates the flight of a full-scale model rocket. The vehicle will be powered by an EDF and equipped with advanced control systems to follow a predetermined trajectory, simulating the flight path of a larger rocket. The team plans to attach a small CanSat payload to the vehicle, which will be deployed at a specific altitude, emulating the tasks required of a full-scale vehicle.

By working on this project, the team hopes to gain hands-on experience in aerospace engineering, specifically in the area of GNC for rockets. The project will provide an opportunity for the team to apply their theoretical knowledge to a practical application, giving them a deeper understanding of how rockets work and the challenges involved in designing and controlling them.

## 1.2 Motivation

The motivation for this project stems from the team's interest in the field of aerospace engineering and space exploration. The team believes that developing a deeper understanding of how rockets work and how they can be controlled will be valuable not only for their personal growth but also for advancing the field of aerospace engineering.

By working on this project, the team hopes to gain practical experience in developing GNC algorithms for a VTOL vehicle that replicates the flight of a full-scale model rocket. This will provide them with a deeper understanding of the challenges involved in designing and controlling rockets, enabling them to apply this knowledge to future projects.

Moreover, the team believes that this project will contribute to the ongoing efforts to enhance space exploration and technology development. By developing more advanced and efficient space vehicles, we can better understand our universe and the challenges of space travel. Additionally, the team believes that this project has the potential to contribute to the development of more efficient and cost-effective payload delivery systems, which could have significant benefits for scientific research and exploration.

## 1.3 Problem statements

The problem statements for above project are given below:

- Rockets are complex systems that require precise control and guidance to ensure safe and efficient operation.
- Existing control systems for space vehicles are often limited in their ability to guide the vehicle along a predetermined trajectory, hindering their efficiency and accuracy.
- There is a need for more efficient and cost-effective payload delivery systems to support scientific research and exploration, which can be achieved by developing more advanced and efficient space vehicles.
- The development of more advanced GNC algorithms for space vehicles is essential for enabling precise control and navigation, ultimately supporting the development of more efficient and cost-effective space vehicles.

## 1.4 Objectives

The objectives of this project are to:

- Develop a control system for a VTOL vehicle using Model Predictive Control Linear Quadratic algorithms.
- Implement the MPC algorithms in a simulated environment and optimize the control system parameters to achieve precise guidance, navigation, and control over a predetermined trajectory.
- Test the LQG control system's performance in a real-world environment, evaluate its effectiveness in enhancing the VTOL vehicle's flight characteristics, and identify areas for future improvement and development.

In summary, the project aims to demonstrate the feasibility and effectiveness of using LQ and MPC algorithms in the design and control of a VTOL vehicle, and to contribute to the ongoing efforts to enhance the capabilities of space vehicles and advance the field of aerospace engineering.

## 1.5 Scope of Project

The project's scope will focus on the implementation of active control systems for precise guidance, navigation, and control over a predetermined trajectory, enabling the vehicle to simulate the flight path of a larger rocket. The vehicle will be powered by an EDF and equipped with advanced control systems.

The team will develop and test these control algorithms in a simulated environment before deploying them on the physical vehicle. They will investigate the challenges associated with developing and testing GNC algorithms for space vehicles, including scalability, precision, and reliability. The project will also assess the limitations of existing control systems and the potential benefits of developing more advanced GNC algorithms.

The project's outcomes will provide valuable insights into the challenges involved in designing and controlling rockets, enabling the team to apply this knowledge to future projects. Moreover, the project will contribute to the ongoing efforts to enhance space exploration and technology development. By developing more advanced and efficient space vehicles, we can better understand our universe and the challenges of space travel.

## 1.6 Application

The GNC algorithms developed for the VTOL vehicle in this project have numerous potential applications in the field of aerospace engineering and space exploration. Some of the potential application areas of the project are:

- **Interplanetary missions:** The advanced control systems developed in this project can enable space vehicles to land more precisely and safely on other planets with dense atmospheres. The GNC algorithms can help to overcome the challenges associated with unpredictable parachute drifts, making space exploration more efficient and cost-effective.

- **Payload delivery systems:** The efficient and precise landing of space vehicles developed in this project can enhance the delivery of payloads, reducing cost and increasing efficiency. This capability can be beneficial in various fields, including scientific research and commercial applications.

- **UAVs**The development of GNC algorithms for the VTOL vehicle can have applications in the design and control of unmanned aerial vehicles, enabling them to navigate more precisely and efficiently in different environments.

- **Defense applications:** The advanced control systems developed in this project can have applications in the design and control of military drones, enabling them to navigate and land more precisely and safely.

- **Commercial applications:** The GNC algorithms developed in this project can be applied in the design and control of commercial drones, enabling them to perform various tasks more efficiently, such as delivery and inspection services.

In summary, the GNC algorithms developed in this project have numerous potential applications in the field of aerospace engineering and space exploration, as well as in other fields, including defense and commercial applications.

# 2. Literature Review

The development of flight computers for use in aerospace applications has become increasingly important with the growth of UAV's and other space exploration technologies. This literature review focuses on the use of Atmega 328p and Teensy microcontrollers for building flight computers, with reference to the works of Pathak et al. (2022) and Sørensen et al. (2017).

Pathak et al. (2022) present the design and implementation of a flight computer for sounding rockets, called the S3FC. The authors describe the use of an Atmega 328p microcontroller as the CPU of the flight computer. The Atmega 328p was chosen due to its low power consumption, low cost, and availability of development tools. The authors describe the various subsystems of the S3FC, including power management, communication, and data storage, and highlight the challenges of designing a flight computer for use in high-altitude and high-speed environments.[1]

Sørensen et al. (2017) describe the use of a Teensy 3.2 microcontroller for building a low-cost and flexible UAV sensor deployment system. The authors describe the various sensors that can be integrated into the system, including cameras, temperature sensors, and air quality sensors. The Teensy 3.2 was chosen for its high processing speed and flexibility, as well as its compatibility with a wide range of sensors and communication protocols. [2]

Overall, the literature suggests that both the Atmega 328p and the Teensy microcontrollers can be effective choices for building flight computers for aerospace applications. The Atmega 328p is a low-cost and low-power option that is well-suited for applications that do not require high processing speeds, while the Teensy offers greater processing power and flexibility for more complex applications. However, both options require careful consideration of the specific requirements of the application, as well as the challenges of designing a flight computer for use in high-altitude and high-speed environments.

Sensor fusion is a process of integrating data from multiple sensors to provide more accurate and reliable information than can be obtained from any single sensor. Sensor fusion has been widely used in navigation, where it is used to combine information from different sensors, such as GPS, magnetometer, accelerometer, and gyroscope, to estimate the position, orientation, and velocity of a moving object. In this literature review, we will discuss recent research on sensor fusion for navigation using GPS, magnetometer, accelerometer, and gyroscope.[3]

GPS is a widely used navigation system that provides accurate position and time information. However, GPS signals can be blocked or degraded by buildings, trees, and other obstacles. To overcome this limitation, researchers have proposed various approaches for integrating GPS with other sensors.[4]

In a study by Rios et al. (2004), the authors proposed a GPS/IMU/magnetometer sensor fusion system for navigation in diverse flight environments.[**?** ] The system uses an Extended Kalman filter to fuse the data from the sensors and estimate the position and orientation of a mobile device. The results showed that the system achieved accurate position and orientation estimation.[5]

Magnetometers are used to measure the magnetic field of the earth, which can be used to estimate the orientation of a mobile device relative to the magnetic north. However, magnetometers are sensitive to external magnetic fields, which can cause measurement errors.

In a study by Kayasal (2007), the authors proposed a magnetometer-aided inertial navigation system for navigation. The system uses a Kalman filter to fuse the data from the magnetometer and the accelerometer to estimate the position and orientation. The results showed that the system achieved accurate position and orientation estimation.[6]

Accelerometers are used to measure the acceleration of a mobile device, which can be used to estimate the velocity and position of the device. However, accelerometers are sensitive to external forces, such as vibrations and shocks.

Gyroscopes are used to measure the angular velocity of a mobile device, which can be used to estimate the orientation and position of the device. However, gyroscopes are sensitive to temperature variations and drift over time.

In a study by Fan et al. (2019), the authors proposed a pedestrian dead reckoning (PDR) system for indoor navigation. The system uses a Kalman filter to fuse the data from the accelerometer and the gyroscope to estimate the position and orientation of a pedestrian. The results showed that the system achieved accurate position and orientation estimation in an pedestrians.[7]

System identification is a powerful technique used to model complex dynamic systems using data-driven approaches. In this literature review, we will focus on the system identification of a Vertical Take-Off and Landing (VTOL) vehicle, specifically for the purpose of yaw, roll, pitch, and altitude control. We will also discuss the use of both blackbox and gray box modeling techniques for this application.

One study conducted by M. Gandhi et al. (2019) used system identification to develop a nonlinear dynamic model of a small unmanned VTOL vehicle. The authors used data collected from a real-world test flight to identify the system parameters, including the aerodynamic coefficients and the motor dynamics.[8]

Another study by Panizza et al. (2015) used a combination of blackbox and gray box modeling techniques to identify the dynamic model of a quadrotor drone for yaw, roll, and pitch control. The authors used a Least Mean Squares algorithm to model the black box part of the system, which included the aerodynamic and motor dynamics. They then used a maximum likelihood estimation approach to estimate the remaining parameters of the model, which were considered to be the graybox part of the system. The resulting model was shown to accurately capture the dynamics of the quadrotor and was used for control design.[9]

In another study, Hann et al. (2010) used system identification to develop a minimal model of rocket roll dynamics and disturbance. The authors analyzed roll dynamics of a sounding rocket in a vertical wind tunnel. A novel method was developed which decoupled the disturbance from the rocket frame's intrinsic roll dynamics and allowed accurate prediction of roll rate and angle. [10]

Therefore, these studies demonstrate the usefulness of system identification for the modeling and control of VTOL vehicles. Both blackbox and gray box modeling techniques can be used, depending on the level of system understanding and the available data. System identification can also be used for different control objectives, including yaw, roll, pitch, and altitude control.

PID Control System has been widely used to in industrial process control and automation systems because they are simple, reliable, and effective in regulating the output of a control system.[11][12] O'Dwyer (2000) summarizes Pi and PID controller tuning rules for time delay processes, which can be particularly relevant for VTOL vehicle control given the inherent delays in the system. The author describes various tuning rules, including the Ziegler-Nichols method and the Cohen-Coon method.[13]

We did thorough reviews on existing research experiments and papers before assuring the design and implementation of Control system, Sensor Fusion and Guidance Navigation of a VTOL for Precision Landing as subject for our final year project. We went through the pros and cons of the methods and algorithms used on the development of MPC, GNC and Sensor fusion so as to come up with efficient and practical implementation for our own project. The problem of robot to a desired position and orientation is considered. In this paper, a model predictive control (MPC) scheme based on tailored non quadratic stage cost is proposed to fulfill this control task. We rigorously prove asymptotic stability while neither stabilizing constraints nor costs are used. To this end, we first design suitable maneuvers to construct bounds on the value function. Second, these bounds are exploited to determine a prediction horizon length such that the asymptotic stability of the MPC closed loop is guaranteed. Finally, numerical simulations are conducted to explain the necessity of having

non quadratic running costs.[14]

Precision landing of a rocket booster or any spacecraft is a relatively new concept. In a paper published in 2015, Pascucci, Bennani  Bemporad[15] discuss how MPC can be used for landing a vehicle to a desired position without pre-configured path, taking into account the full system dynamics and actuators and state constraints. They first predicted the linear model of the nonlinear system by white box system identification method and used MPC for thrust vectoring control to change the magnitude and direction of force generated by the model's engines. This research is the base for using model predictive control techniques in precision landing. The paper describes landing within hundred meters from a target as a future challenge. We have come a long way since then and rockets have been successfully landed even on aircraft carriers.

# 3. Related theory

## 3.1 Harware and components

### 3.1.1 Micro-controller Contrasting

The parameters of microcontrollers used i.e. Atmega 328P and Teensy 4.1 are given below:

**Clock frequency:**

The Teensy 4.1 has a much higher maximum clock frequency of 600 MHz, compared to the ATmega328P's maximum clock frequency of 20 MHz.

**Number of pins:**

The Teensy 4.1 has 40 pins, while the ATmega328P has 28 pins.

**Digital pins and analog pins:**

The Teensy 4.1 has 34 digital pins and 18 analog pins, while the ATmega328P has 14 digital pins and 6 analog pins.

**Working voltage:**

The ATmega328P can operate at a voltage range of 1.8V to 5.5V, while the Teensy 4.1 can operate at a voltage range of 3.3V to 6.0V.

**Program memory:**

The Teensy 4.1 has 2MB of Flash program memory, while the ATmega328P has 32KB of Flash program memory.

**PWM pins:**

The Teensy 4.1 has 22 PWM pins, while the ATmega328P has 6 PWM pins.

**Interrupt pins:**

The Teensy 4.1 has up to 26 interrupt pins, while the ATmega328P has two external interrupt pins, INT0 and INT1.

**I2C:**

Both the ATmega328P and Teensy 4.1 have dedicated I2C pins, SCL and SDA . However, on the Teensy 4.1 they are labeled SCL1 and SDA1.

**Hardware interrupt pins:**

Both the ATmega328P and Teensy 4.1 have hardware interrupt pins, with the Teensy 4.1 having up to 26 interrupt pins.

**Hardware serial pins:**

The ATmega328P has one hardware serial port, while the Teensy 4.1 has up to 8 hardware

serial ports.

Overall, the Teensy 4.1 has significantly higher performance than the ATmega328P in terms of clock frequency, program memory, and number of digital and analog pins. It also has more PWM and interrupt pins, as well as more hardware serial ports, which can be useful for real-time control and data processing. Additionally, both microcontrollers have I2C pins for communication with other devices, but on the Teensy 4.1 they are labeled differently. The ATmega328P has a wider working voltage range, which can make it more versatile in certain applications.

### 3.1.2 Flight Computer Components

MPU6050

The MPU6050 is a popular 6-axis motion tracking sensor that combines a 3-axis gyroscope and a 3-axis accelerometer on a single chip. It is commonly used for orientation sensing and motion tracking. The MPU6050 has programmable full-scale range settings for both the accelerometer and gyroscope. The accelerometer can be set to ±2g, ±4g, ±8g, or ±16g, while the gyroscope can be set to ±250°/s, ±500°/s, ±1000°/s, or ±2000°/s. These settings determine the maximum measurable acceleration or angular velocity of the device. The MPU6050 also has a Digital Low Pass Filter (DLPF) that can be used to reduce noise and improve stability in the sensor readings. The DLPF can be programmed with different settings, including a bandwidth of 260Hz for the gyroscope and 256Hz for the accelerometer, or a bandwidth of 5Hz for both sensors. Choosing the appropriate DLPF setting depends on the specific application and the desired trade-off between responsiveness and noise reduction.

The MPU6050 also includes an internal Digital Motion Processor (DMP) that can perform complex calculations on the raw sensor data to provide sensor fusion and motion processing. The DMP can output quaternion data, which represents the orientation of the sensor in 3D space, as well as other useful information such as gravity-compensated linear acceleration and heading angle. The DMP can run at a maximum sampling rate of 1000Hz, making it useful for real-time motion tracking applications. Overall, the combination of the gyroscope, accelerometer, and DMP make the MPU6050 a powerful and versatile sensor for motion tracking and orientation sensing.

MPU9250

The MPU9250 is another popular 9-axis motion tracking sensor, which includes a 3-axis gyroscope, 3-axis accelerometer, and 3-axis magnetometer on a single chip. It is an upgraded version of the MPU6050, with additional magnetometer functionality. The MPU9250 offers similar programmable full-scale range settings for the accelerometer and gyroscope as the MPU6050, with the accelerometer ranging from ±2g to ±16g and the gyroscope ranging from

±250°/s to ±2000°/s. However, the magnetometer can be set to a full-scale range of ±4800T, which is much higher than the typical range of ±2.5 Gauss seen in most magnetometers. The MPU9250 also includes a Digital Low Pass Filter (DLPF) for noise reduction, which can be programmed with different bandwidth settings. The DMP in the MPU9250 is more advanced than the one in the MPU6050, and it can perform sensor fusion on all three sensors (gyroscope, accelerometer, and magnetometer) to provide highly accurate and stable motion tracking data. The DMP in the MPU9250 can output quaternion data and other useful information, such as calibrated sensor data and heading angle, at a maximum sampling rate of 1000Hz.

BMP280

The BMP280 features a high-precision MEMS pressure sensor that can measure atmospheric pressure with an accuracy of up to ±1 hPa, and a temperature sensor with an accuracy of up to ±1°C. The sensor can operate over a wide temperature range of -40°C to 85°C, making it suitable for use in harsh environments. The BMP280 is a digital sensor that communicates over I2C and SPI interfaces, and it can be configured with different modes of operation to optimize power consumption and accuracy. It also features an onboard 16-bit ADC (Analog to Digital Converter) that can convert the analog sensor readings into digital data.

Lidar

The TFmini Lidar uses a pulsed laser diode and an optical receiver to measure the distance between the sensor and an object in its field of view. It can accurately detect objects at a distance of up to 12 meters, with a resolution of 5mm. The sensor has a wide field of view of 2.3 degrees, allowing it to capture data from a broad area in front of the sensor. The TFmini Lidar communicates with a host device over a UART interface.

GPS Module

The NEO-M8N GPS module features a 72-channel GPS receiver that can receive signals from multiple GPS satellite systems, including GPS, GLONASS, Galileo, and BeiDou. This allows the module to provide highly accurate positioning data with a positional accuracy of up to 2.5 meters. The module communicates with a host device over a UART interface and supports a range of standard GPS protocols such as NMEA, UBX, and RTCM. It also features a built-in data logger that can store up to 16 hours of GPS data for later analysis.

NRF24L01

The NRF24L01 is a transceiver uses the 2.4GHz ISM band and provides a range of up to 100 meters in open space, with a data transfer rate of up to 2Mbps. The module features a built-in antenna, and it communicates with other devices over a SPI interface. The NRF24L01 supports a range of advanced features, such as automatic packet retransmission, dynamic channel selection, and multi-level data pipe support. It can act as both a transmitter and

receiver

## 3.2 Sensor Fusion

### 3.2.1 MPU6050

The MPU-6050 is a popular integrated circuit that combines a 3-axis gyroscope and a 3-axis accelerometer in a single package, along with an on-board Digital Motion Processor (DMP) that can perform sensor fusion to output quaternions or Euler angles.

The MPU-6050 has a programmable low-pass filter for both the gyroscope and accelerometer. The filter is implemented using a 2nd-order Infinite Impulse Response (IIR) filter, which is defined by the following difference equation:

$$y[n] = b_0x[n] + b_1x[n-1] + b_2x[n-2] - a_1y[n-1] - a_2y[n-2] \tag{3.1}$$

where $x[n]$ is the input signal, $y[n]$ is the output signal, $b_0$, $b_1$, $b_2$ are the filter coefficients for the input signal, and $a_1$, $a_2$ are the filter coefficients for the output signal.

The low-pass filter cutoff frequency is determined by the filter coefficients, which are programmed using the MPU-6050 registers. The gyroscope and accelerometer low-pass filters can be programmed independently with different cutoff frequencies.

The MPU-6050 also has an on-board Digital Motion Processor (DMP) that can perform sensor fusion to output quaternions or Euler angles. The DMP fuses the gyroscope, accelerometer, and magnetometer (if available) data to estimate the device orientation.

The DMP uses a complementary filter to combine the gyroscope and accelerometer data. The complementary filter is defined by the following equation:

$$q_{n+1} = \frac{1}{\sqrt{2}} \left[ \mathbf{q}_n \otimes \omega_n \Delta t \;\; 1 - \tfrac{1}{2}|\omega_n|^2 \Delta t^2 \right] \left[ \mathbf{q}_n \;\; \tfrac{1}{\sqrt{2}} \right] \tag{3.2}$$

where $q_{n+1}$ is the estimated quaternion at time $n+1$, $\mathbf{q}_n$ is the estimated quaternion at time $n$, $\omega_n$ is the angular velocity measurement at time $n$, $\Delta t$ is the time step, $\otimes$ denotes quaternion multiplication, and $|\cdot|$ denotes the Euclidean norm.

The DMP also includes a 6-axis Kalman filter for estimating the accelerometer and gyroscope biases, as well as a magnetometer calibration algorithm for estimating the magnetometer biases and scale factors.

### 3.2.2 MPU9250

The MPU 9250 is an upgraded version of the MPU 6050 and includes additional sensors such as a magnetometer, making it capable of 9-axis motion tracking. In terms of the accelerometer and gyroscope performance, both sensors have similar ranges and sensitivities.

One major difference between the two sensors is the implementation of the Digital Motion Processor (DMP). The MPU 6050 has an internal DMP which allows for the offloading of sensor fusion calculations from the host microcontroller. The DMP includes features such as low pass filters and the ability to calculate quaternions for orientation estimation.

The MPU 9250 also includes an internal DMP, but with additional features such as support for the magnetometer data and a higher output data rate of up to 1000 Hz. The DMP on the MPU 9250 also includes the ability to perform motion analysis, such as detecting tap and shake gestures.

Overall, the MPU 9250 provides additional functionality and performance improvements over the MPU 6050, but at a higher cost.

### 3.2.3 Madgwick Filter

The Madgwick filter is a popular algorithm for sensor fusion that is commonly used in low-power embedded systems due to its computational efficiency. The filter uses a quaternion-based representation of orientation, similar to the extended Kalman filter.

The Madgwick filter is based on a gradient descent optimization method, which minimizes the error between the measured sensor data and the predicted values based on the estimated orientation. The filter estimates the orientation of an object using a combination of accelerometer, gyroscope, and magnetometer data.

The filter state is represented by a quaternion $q_k$, which describes the orientation of the object in space. The update equation for the quaternion is given by:

$$q_{k+1} = q_k \otimes \exp\left(\frac{\Delta t}{2} \cdot \omega_{bias}\right) \otimes \exp\left(\frac{\Delta t}{2} \cdot \omega_k\right) \tag{3.3}$$

where $\otimes$ denotes the quaternion multiplication operator, $\omega_{bias}$ is the gyroscope bias, $\omega_k$ is the angular velocity measurement, and $\Delta t$ is the time step.

The filter also estimates the gyroscope bias and uses it to correct for drift in the gyroscope measurements. The gyroscope bias estimate is updated using the equation:

$$\omega_{bias} = \omega_{bias} + \beta \cdot \sum_{i=1}^{n} J_{ij} \cdot F_i \tag{3.4}$$

where $n$ is the number of sensors, $J_{ij}$ is the Jacobian matrix of the $i$-th sensor measurement with respect to the $j$-th component of the quaternion, $F_i$ is the error between the predicted and measured values for the $i$-th sensor, and $\beta$ is a tuning parameter.

The Madgwick filter also incorporates a magnetic distortion correction algorithm, which corrects for the effects of magnetic interference on the magnetometer measurements. The corrected magnetometer measurement is given by:

$$m_k = C_m(m_{raw} - b_m) \tag{3.5}$$

where $m_{raw}$ is the raw magnetometer measurement, $b_m$ is the magnetometer bias, and $C_m$ is the correction matrix.

### 3.2.4 Extended Kalman Filter

The extended Kalman filter (EKF) is a method for estimating the state of a dynamic system based on noisy sensor measurements. The state vector, denoted by $x_k$, includes the position, velocity, and orientation of the object at time $k$, represented by $p_k$, $v_k$, and $q_k$ (a quaternion), respectively. The dynamic model used in the EKF is based on the kinematic equations of motion, which relate the acceleration, velocity, and position of an object. The state propagation equation predicts the state estimate at the next time step, denoted by $\hat{x}_k^-$:

$$\hat{x}k^- = f(xk-1, u_k, \Delta t) \tag{3.6}$$

where $u_k$ is the control input, and $\Delta t$ is the time step.

Each sensor is associated with a measurement model that relates the sensor measurement $z_k$ to the state vector $x_k$ through a measurement function $h_k$ and measurement noise $v_k$:

$$z_k = h_k(x_k) + v_k \tag{3.7}$$

The EKF consists of two steps: the prediction step and the update step. The prediction step uses the state propagation equation to predict the state estimate at the next time step:

$$\hat{x}k^- = f(\hat{x}k-1, u_k, \Delta t) \tag{3.8}$$

where $\hat{x}_{k-1}$ is the previous state estimate.

The update step corrects the predicted state estimate using the measurement data:

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - h_k(\hat{x}_k^-)) \tag{3.9}$$

where $K_k$ is the Kalman gain, which is computed as:

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \tag{3.10}$$

where $P_k^-$ is the predicted state covariance matrix, $H_k$ is the measurement Jacobian matrix, and $R_k$ is the measurement noise covariance matrix.

The state covariance matrix is updated as:

$$P_k = (I - K_k H_k) P_k^- \tag{3.11}$$

where $I$ is the identity matrix.

The EKF can be implemented asynchronously to efficiently process multiple sensors with different update rates. Additionally, algorithms can be used to handle sensor failures and initialize the filter when the sensor data is not available.

## 3.3 System Modeling

### 3.3.1 First Principal Approach

During the earlier phase of our project, we used a white-box modeling approach, also known as the physics-based approach, for identifying the mathematical model of our system. This approach involves deriving a model based on the physical laws and properties of the system. We derived a simple mathematical model for pitch, roll, and altitude control of our rocket model based on Newton's laws. However, we found that the simplistic model was not able to fully define our system, and many dynamics of the system remained unidentified.

While the white-box modeling approach is a powerful technique for modeling a system based on physical laws, it has limitations. The approach relies on accurate knowledge of the system's physical properties, and if these properties are not well understood or if there are other complex dynamics involved, the model may not be accurate or complete.

### 3.3.2 Data Driven Approach

Data-driven approaches, also known as black-box modeling or system identification, are a class of techniques used to derive mathematical models of complex systems based solely on measured input-output data. Unlike the white-box modeling approach, which relies on knowledge of the physical laws and properties of the system, data-driven approaches aim to infer the underlying dynamics of the system directly from observed input-output data.

System identification is the process of using data-driven techniques to build mathematical models of a system, such as transfer functions, state-space models, or other types of models. These models can then be used for analysis, simulation, and control of the system. Data-driven approaches are particularly useful when the physical laws and properties of a system are not well understood, or when the system is too complex to be modeled using white-box techniques alone. Data-driven approaches can also be used to supplement white-box models by identifying additional dynamics that are not captured by the physical laws.

Common techniques used in system identification include subspace methods, time-series analysis, and parametric modeling techniques such as ARX and ARMAX models. These techniques allow the identification of system dynamics, including the relationship between inputs and outputs, and can be used to derive models suitable for control and analysis purposes. Overall, data-driven approaches provide a powerful tool for identifying mathematical

models of complex systems, allowing for accurate analysis, simulation, and control of these systems.

### 3.3.3 Linear and Non Linear models

System identification techniques can be used to derive both linear and non-linear models of a system. Linear models are widely used in control theory and signal processing and are characterized by a linear relationship between the system inputs and outputs. These models are typically described by transfer functions or state-space models and can be identified using frequency response analysis or time-domain analysis techniques.

On the other hand, non-linear models are characterized by non-linear relationships between the system inputs and outputs, making them more complex and challenging to identify. Non-linear models are often required for modeling complex systems such as biological systems, chemical processes, and robotics, where the relationships between the inputs and outputs are highly non-linear. System identification techniques can be used to identify non-linear models, which can be useful for analysis, simulation, and control of these systems.

There are several approaches for identifying non-linear models, including the use of parametric models, neural networks, fuzzy logic systems, and kernel methods. These techniques involve estimating the parameters of the non-linear model from the input-output data, and can be used to construct a model that accurately captures the system's behavior.

In general, linear models are simpler and easier to analyze than non-linear models. However, non-linear models can be more accurate and better suited for modeling complex systems. System identification techniques can be used to derive both linear and non-linear models of a system, depending on the complexity of the system and the accuracy required for analysis, simulation, and control.

While our rocket model is governed by non-linear dynamics, we chose to use a linear model for control design due to its simplicity and ease of use with control techniques such as LQR and MPC. The linear model was obtained by approximating the non-linear system dynamics around the control values of interest.

While a non-linear model may have better represented the behavior of our system, the linear approximation allowed us to design controllers that were effective at stabilizing the system and achieving the desired control objectives. Additionally, the linear model was simpler to work with, allowing for more straightforward analysis and implementation. It is important to note, however, that the use of a linear model may have limitations in accurately capturing the full behavior of our system. In future work, it may be worthwhile to explore the use of non-linear models for control design, especially in cases where the system behavior is highly non-linear and the control objectives are more complex. Overall, our decision to

use a linear model for control design was motivated by the desire for simplicity and ease of use with existing control techniques. While a non-linear model may have provided a more accurate representation of our system, the linear model was sufficient for achieving our control objectives and provided a useful starting point for further analysis and exploration.

### 3.3.4 Time Invariance

Time invariance is a fundamental property of a system that describes how the behavior of the system remains constant over time. A system is said to be time-invariant if its response to a given input signal is the same regardless of when the input is applied. This means that if an input signal is delayed or advanced in time, the output signal will also be delayed or advanced in time by the same amount.

In mathematical terms, a system is time-invariant if and only if its impulse response is the same for all time intervals.

The time invariance property is closely related to the principle of superposition, which states that the response of a linear system to a sum of input signals is equal to the sum of the individual responses of the system to each input signal.

### 3.3.5 Block diagram of system for system identification

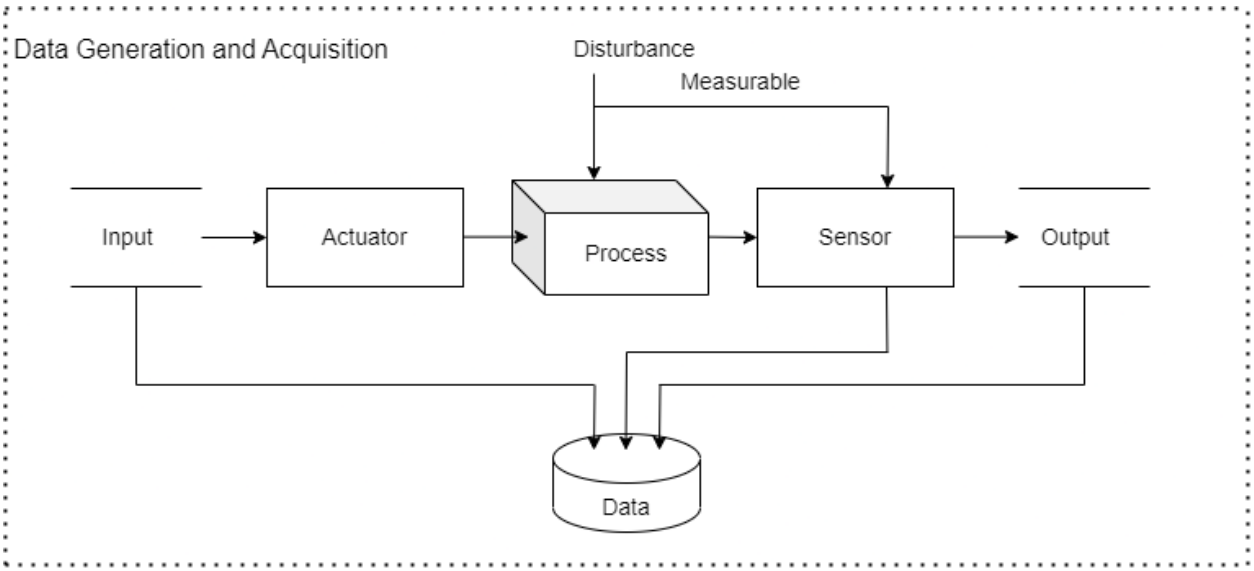



Figure 3.1: Block diagram of system for system identification

In order to achieve proper hovering and trajectory following of our rocket model, it is essential to have precise control over its movement in all four directions - yaw, pitch, roll, and altitude. To achieve this level of control, we need to develop a mathematical model of the system that accurately describes its behavior in each of these dimensions.

For instance, the yaw control requires a model that describes the movement of the rocket around the vertical axis, while pitch control requires a model that describes the movement around the lateral axis, and roll control requires a model that describes the movement around the longitudinal axis. Additionally, altitude control requires a model that describes the movement of the rocket in the vertical direction.

### 3.3.6 ARX Method For System Identification For SISO system

The ARX method is a commonly used technique for estimating the transfer function of a linear time-invariant system from input-output data. The transfer function can be represented as a ratio of two polynomials:

$$G(s) = b0 + b1s^(-1) + ... + bms^(-m)/a0 + a1s^(-1) + ... + ans^(-n) \tag{3.12}$$

where s is the complex frequency variable, and the coefficients **b0, b1, ..., bm** and **a0, a1, ..., an** are unknown parameters to be estimated from input-output data. The ARX model is used to approximate the true transfer function by fitting a polynomial ratio to the input-output data. The model can be written as:

$$y(t) = b0u(t) + b1u(t-1) + ... + bmu(t-m) - a1y(t-1) - ... - any(t-n) \tag{3.13}$$

where **y(t)** is the output of the system at time **t, u(t)** is the input to the system at time **t**, and **m** and **n** are the orders of the numerator and denominator polynomials, respectively. The ARX method involves estimating the coefficients of the numerator and denominator polynomials that best fit the input-output data. This is typically done using least squares regression, where the sum of squared errors between the predicted output and the actual output is minimized:

$$minimizesum_i(y(i) - y_hat(i))^2 \tag{3.14}$$

where $y_hat(i)$ is the predicted output at time i based on the model parameters and the input data.

Once the coefficients of the numerator and denominator polynomials have been estimated, the transfer function can be calculated as:

$$G(s) = Y(s)/U(s) \tag{3.15}$$

where Y(s) and U(s) are the Laplace transforms of the output and input signals, respectively, and are given by:

$$Y(s) = b0U(s) + b1U(s)s^(-1) + ... + bmU(s)s^(-m) \tag{3.16}$$

$$U(s) = 1/(a0 + a1s^( - 1) + ... + an * s^( - n)) \tag{3.17}$$

In summary, the ARX method is a commonly used technique for estimating the transfer function of a linear time-invariant system from input-output data. The method involves fitting an ARX model to the input-output data and estimating the coefficients of the numerator and denominator polynomials using least squares regression. Once the coefficients have been estimated, the transfer function can be calculated as a ratio of two polynomials.

### 3.3.7 ARMAX Method for System Identification SISO system

ARMAX (AutoRegressive Moving Average with eXogenous inputs) model is a method of system identification used to model the input-output relationship of a dynamic system with external inputs. ARMAX models build upon ARMA models by adding exogenous input terms. ARMAX models can be written in the following form:

$$A(q)y(t) = B(q)u(t - d) + C(q)e(t) \tag{3.18}$$

where $\mathbf{y(t)}$ is the output of the system, $u(t - d)$ is the delayed input to the system, $e(t)$ is the white noise disturbance, and $A(q)$, $B(q)$, and $C(q)$ are polynomials in the lag operator $\mathbf{q^{-1}}$.

The ARMAX model is specified by three parameters:

1. The order of the AR polynomial, denoted by $n_a$.
2. The order of the MA polynomial, denoted by $n_c$.
3. The order of the exogenous input polynomial, denoted by $n_b$.

The ARMAX model can be estimated from input-output data using least squares or maximum likelihood methods. The transfer function of the ARMAX model can be obtained by taking the ratio of the output polynomial to the input polynomial, which gives:

$$G(q) = \frac{B(q)}{A(q)}$$

The variance of the white noise can be estimated from the residual obtained after fitting the ARMAX model to the data. The residuals are the difference between the actual output and the predicted output based on the ARMAX model. The variance of the residual can be used to estimate the variance of the white noise. The ARMAX model is a powerful tool for system identification as it allows the user to model the effects of exogenous inputs on the system. This makes it useful for modeling and controlling systems with external disturbances.

However, it is important to note that ARMAX models are computationally intensive and may require significant computational resources for large systems. Additionally, the accuracy of the ARMAX model is dependent on the quality and quantity of the input-output data used for Estimation.

### 3.3.8 Subspace method for system identification for MIMO system

Subspace identification methods are a class of model-based approaches for system identification that can accurately estimate the state-space model of a linear time-invariant system from input-output data. The MOESP (Multivariable Output Error State-space) algorithm is a subspace identification method for continuous-time systems with a noise model and has been used for the MIMO system identification of our rocket model.

The MOESP algorithm works by first constructing a block Hankel matrix from the input-output data, which is then decomposed using singular value decomposition (SVD) to extract the state and output subspaces of the system. The state and output subspaces are used to estimate the system matrices, which are then combined to form the state-space model.

Mathematically, the MOESP algorithm can be described as follows. Consider a state-space model of the form:

$$\begin{aligned} \frac{dx}{dt} &= Ax + Bu \\ y &= Cx + Du \end{aligned}$$

where $x$ is the state vector, $u$ is the input vector, $y$ is the output vector, and $A$, $B$, $C$, and $D$ are the system matrices to be estimated. Let $Y(t) = [y(t), y(t+T), ..., y(t+(N-1)T)]$ be the output data, and $U(t) = [u(t), u(t+T), ..., u(t+(N-1)T)]$ be the input data, where $T$ is the sampling time and $N$ is the number of samples.

The MOESP algorithm constructs the following block Hankel matrix:

$$H = \begin{bmatrix} Y(0) & Y(T) & \dots & Y((n-1)T) \, U(T) & U(2T) & \dots & U(nT) \end{bmatrix}$$

where $n$ is the number of columns in $H$. Next, we perform the SVD of the Hankel matrix:

$$H = U \cdot S \cdot V^T$$

where $U$ and $V$ are orthonormal matrices, and $S$ is a diagonal matrix containing the singular values of $H$. The left singular vectors of $U$ corresponding to the largest singular values are used to estimate the state subspace of the system:

$$X = \begin{bmatrix} U(:,1) & U(:,2) & \dots & U(:,n-1) \end{bmatrix}$$

where $X$ is an $(n-1) \times n$ matrix. The output subspace of the system is estimated as:

$$Y = \begin{bmatrix} CX & CAX & \dots & CA^{n-2}X \end{bmatrix}$$

where $Y$ is an $m \times ((n-1)p)$ matrix, $p$ is the number of inputs, and $m$ is the number of outputs. The system matrices are then estimated by solving the following optimization problem:

$$\min_{D} \|Y - D \cdot U\|_F^2 \quad \text{subject to} \quad X^T X = I$$

where $D$ is a matrix that combines $C$ and the estimated state-space matrices, and $\|.\|_F^2$ is the Frobenius norm. The optimization problem can be solved using least-squares or other numerical methods, which yields the estimated system matrices $A$, $B$, $C$, and $D$.

To account for measurement noise, the MOESP algorithm can be extended to include a noise model. The noise model assumes that the output measurements are corrupted by Gaussian white noise with zero mean and known covariance matrix $R$. The noise model is incorporated into the Hankel matrix by adding an extra block:

$$H = \begin{bmatrix} Y(0) & Y(T) & \dots & Y((n-1)T) & U(T) & U(2T) & \dots & U(nT) & N(0) & N(T) & \dots & N((n-1)T) \end{bmatrix}$$

where $N(t)$ is the noise vector at time $t$.

The optimization problem for the MOESP algorithm with noise model is then modified to include the noise model:

$$\min_{D,G} \|Y - D \cdot U - G \cdot N\|_F^2 \quad \text{subject to} \quad X^T X = I$$

where $G$ is a matrix that combines the estimated system matrices and the noise covariance matrix $R$, and $\|.\|_F^2$ is the Frobenius norm.

The optimal estimate of the system matrices can be obtained by solving the following generalized least-squares problem:

$$[\hat{A}, \hat{B}, \hat{C}, \hat{D}, \hat{K}] = \arg\min_{D,G} \|Y - D \cdot U - G \cdot N\|_F^2 \qquad \text{subject to} \quad X^T X = I$$

where $\hat{K}$ is the estimated Kalman gain matrix that minimizes the mean squared error between the estimated state and the true state.

The MOESP algorithm with noise model is a powerful technique for identifying continuous-time systems from noisy input-output data. It can accurately estimate the system matrices, even in the presence of measurement noise, and has been successfully applied in various engineering applications, such as control system design, fault detection and diagnosis, and process optimization.

In summary, the MOESP algorithm for continuous-time system identification with noise model involves the following steps:

1. Construct a block Hankel matrix from the input-output data.
2. Perform SVD on the Hankel matrix to extract the state and output subspaces.
3. Estimate the system matrices by solving a least-squares optimization problem.
4. Incorporate a noise model into the optimization problem to account for measurement noise.
5. Solve a generalized least-squares problem to obtain the optimal estimate of the system matrices and the Kalman gain matrix.

The mathematical details of the MOESP algorithm with noise model can be included in a project report to provide a clear and detailed explanation of the methodology used for system identification.

## 3.4 Controller

Control systems have recently gained a more significant part in the growth and development of technology. Almost all facets of our daily actions are influenced by some sort of control system. All areas of industry use automatic control systems in some capacity, including quality control of produced goods, automatic assembly lines, machine-tool control, space technology and weaponry, computer control, power systems, robotics, and many more. In the process industries, it is crucial for tasks like pressure, temperature, humidity, and flow management. Modern control theory has recently been applied to non-engineering systems like biological, biomedical, inventory control, economic, and socioeconomic systems.

All controllers have a specific use case to which they are best suited. We cannot just insert any type of controller at any system and expect a good result – there are certain conditions that must be fulfilled.
The fundamental components of a control system are as follows:

- Objectives of control.
- Control system components.
- Results or output.


Figure 3.2: Control Block

Types of Controller:

- Open Loop Controller
- Close Loop Controller

### 3.4.1 Open Loop Controller:

A system in which the output has no effect upon the input quantity is called open-loop control system. Or simply any system which cannot control or correct the variation on Output.

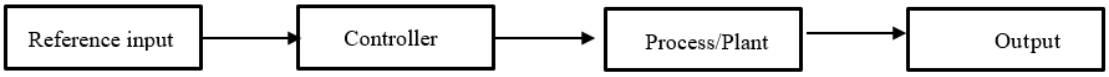



Figure 3.3: Open Loop Block

### 3.4.2 Close loop control system:

The control system in which the output has an effect upon the input quantity so as to maintain the desired output is called closed-loop control system. Or one that measures its output and adjusts its input accordingly by using a feedback signal.

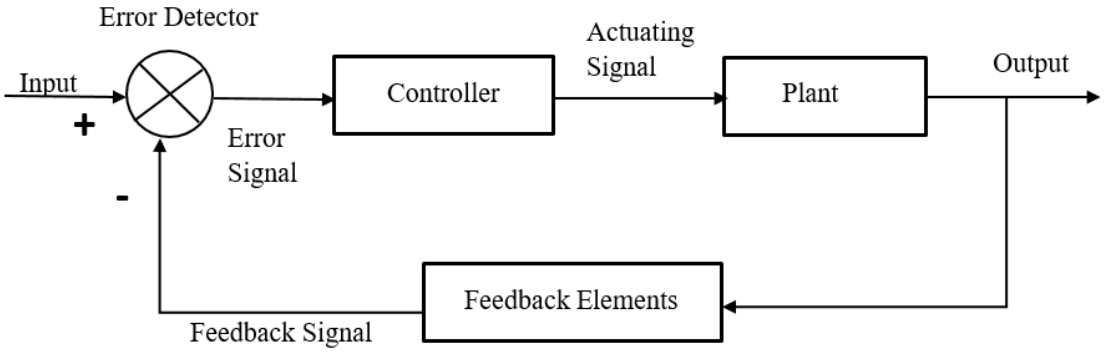



Figure 3.4: Close Loop Block

Type of controller to use must be decided depending upon the nature of the plant and the operating condition, including such consideration as safety, cost, availability, reliability,

accuracy, weight and size

### 3.4.3 Proportional Integral Derivative (PID) Controller

The term PID stands for proportional integral derivative and it is one kind of device used to control different process variables. It produces an output signal consisting of three terms –one proportional to error, another one proportional to integral of error signal and third one proportional to derivative of error signal.

A closed-loop system like a PID controller includes a feedback control system. This system evaluates the feedback variable using a fixed point to generate an error signal. Based on that, it alters the system output. This procedure will continue till the error reaches Zero otherwise the value of the feedback variable becomes equivalent to a fixed point.

The proportional controller stabilizes the gain but produces a steady state error. The integral controller reduces or eliminates the steady state error. The derivative controller reduces the rate of change of error.

- PID controller has higher stability.
- It has no offset.
- It has reduced overshoot.

With the PID control action; there is no offset, no oscillations with least setting time. So there is improvement in both transient as well as steady state response.

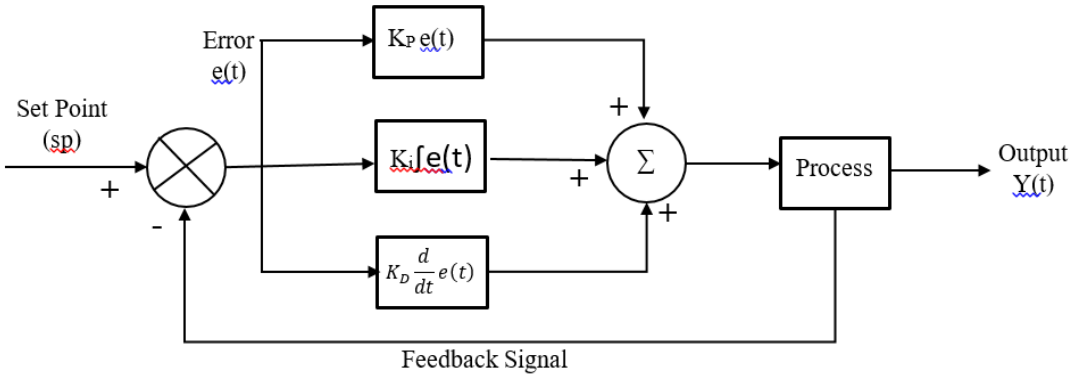



Figure 3.5: PID Block Diagram

**Mathematically:**

$$Y(k)\alpha e(k) + \int e(k) + \frac{d}{dt}e(k) \tag{3.19}$$

$$Y(k) = K_P e(k) + K_I \int e(k) + K_D \frac{d}{dt} e(k) \tag{3.20}$$

**Effect of P, I & D transient response**

| Action | Rise | Overshoot | Settling time | Steady state error |
|---|---|---|---|---|
| Kp | Decrease | Increase | small change | Decrease |
| Ki | Decrease | Increase | Initially decrease then increase | Eliminate |
| Kd | small change | Decrease | Decrease | small change |

Figure 3.6: Effect of different parameter's of PID

### 3.4.4 Linear Quadratic Regulator (LQR):

It is well-known method that provides optimally controlled feedback gains to enable the closed-loop stable and high performance design of system. The LQR algorithm is essentially an automated way of finding an appropriate state-feedback controller. The LQR algorithm reduces the amount of work done by the control system engineer to optimize the controller. However the engineer still need to specify the cost function parameters, and compare the result with the specified design goals. Often this means that controller construction will be an iterative process in which the engineer judges the "optimal" controllers produced through simulation and then adjusts the parameters to produce a controller more consistent with design goal.

### 3.4.5 MPC:

MPC works by using a mathematical model of the system being controlled to predict future behavior, and then optimizing control actions based on these predictions. In the case of the VTOL prototype, the MPC algorithm uses a model of the VTOL to predict how the VTOL will behave in response to control inputs, such as changes in EDF motor thrust or servo tilt angle.

To implement MPC on the VTOL prototype, we that includes sensors like MPU9250, BMP280, GPS, LIDER to measure the VTOL position, velocity, and other parameters, as well as actuators to adjust the EDF motor thrust and Servo tilt angle. The MPC algorithm

runs on a computer, which receives input from the sensors and uses the model to predict how the VTOL will behave.

The MPC algorithm then calculates the optimal control inputs to achieve a desired trajectory, taking into account constraints such as the maximum thrust and the maximum tilt angle. These control inputs are sent to the actuators, which adjust the thrust and tilt angle in real-time.

The use of MPC on the VTOL prototype has several advantages. First, it allows for precise control of the VTOL trajectory, which is essential for achieving the desired flight path. Second, it can adapt to changing conditions, such as wind or other disturbances, to ensure that the VTOL stays on course. Finally, it can handle constraints such as the maximum thrust or tilt angle, ensuring that the rocket operates safely and efficiently and versatility of this control strategy, and its potential to improve the performance and safety of complex systems like VTOL's.

**Single Input Single Output:**

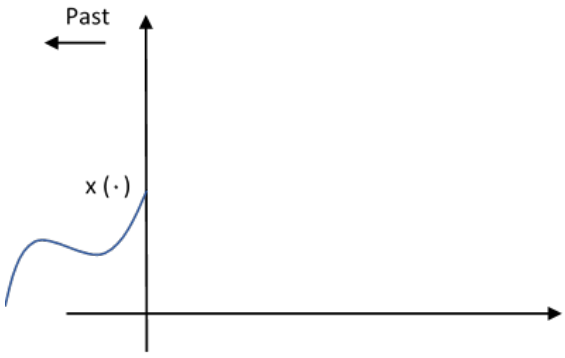
$$x(k+1) = f(x(k), u(k)) \tag{3.21}$$

At decision Instant k, measure the state x (k)

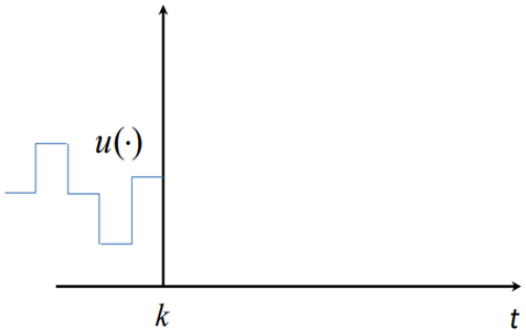
Based on x (k), compute the optimal sequence of control over a prediction horizon N:

$$u^*(x(k)) = (u^*(k), u^*(k+1), \ldots\ldots u^*(k+N-1) \tag{3.22}$$

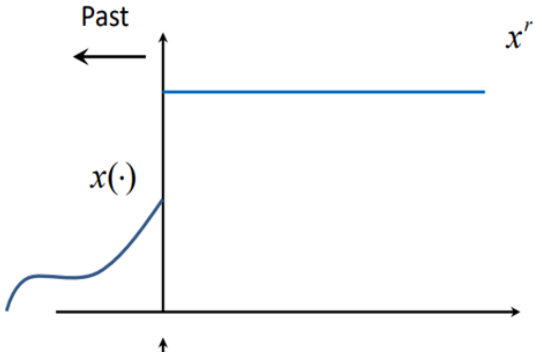
Apply the control u*)(k) on the sampling period [k, k+1].



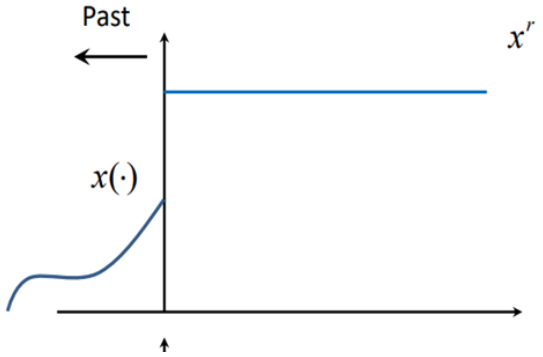

(a)



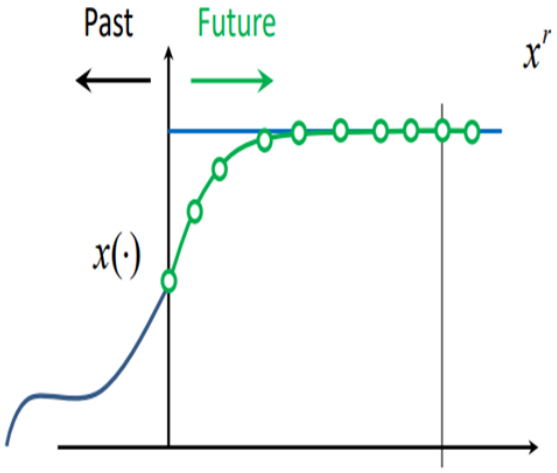

(b)



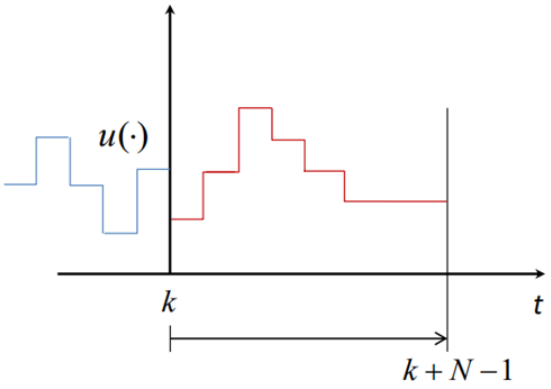

(c)



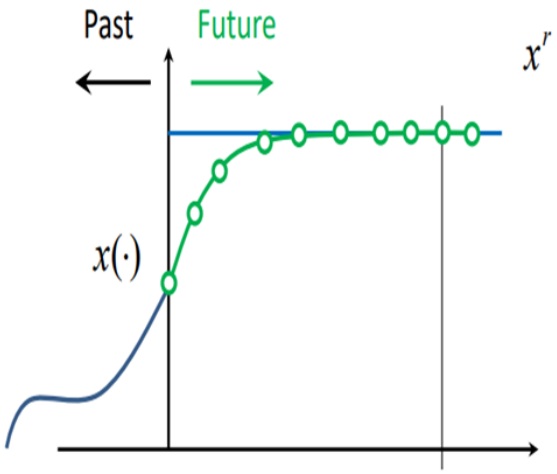

(d)



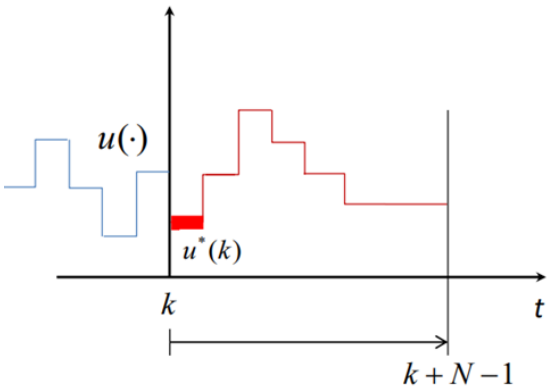

(e)



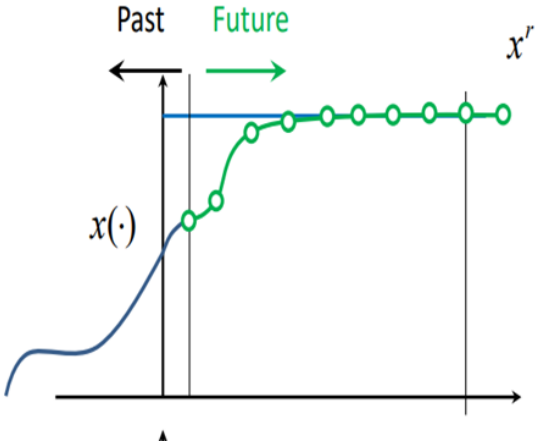

(f)


Past
Future
$x^r$
$x(\cdot)$

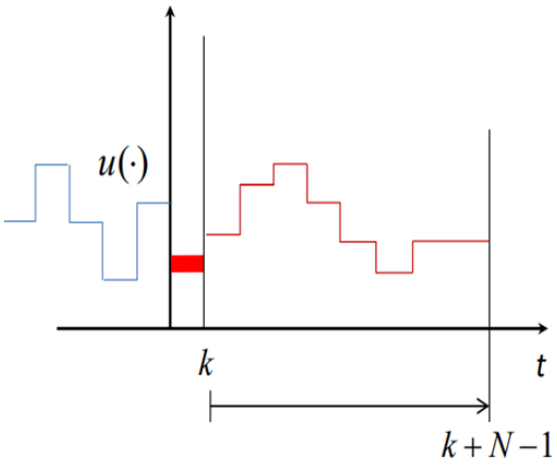
(a)


$u(\cdot)$
$u^*(k)$
$k$
$t$
$k+N-1$

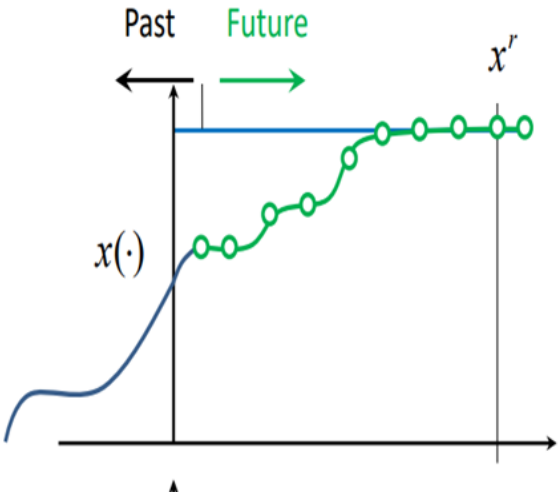
(b)


Past
Future
$x^r$
$x(\cdot)$

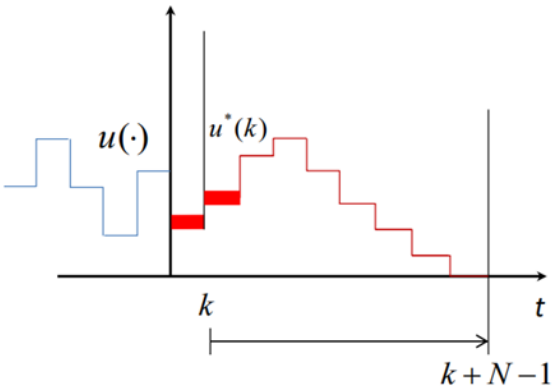
(c)


$u(\cdot)$
$k$
$t$
$k+N-1$

(d)


Past
Future
$x^r$
$x(\cdot)$

(e)

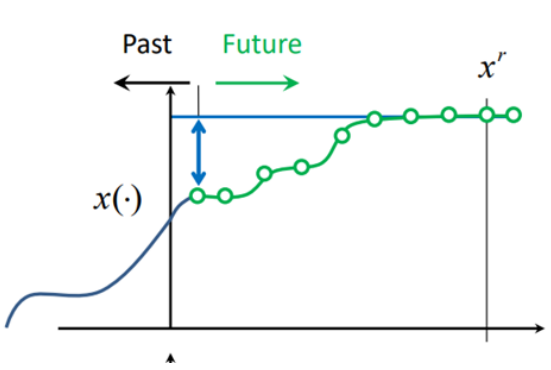

$u(\cdot)$
$u^*(k)$
$k$
$t$
$k+N-1$

(f)

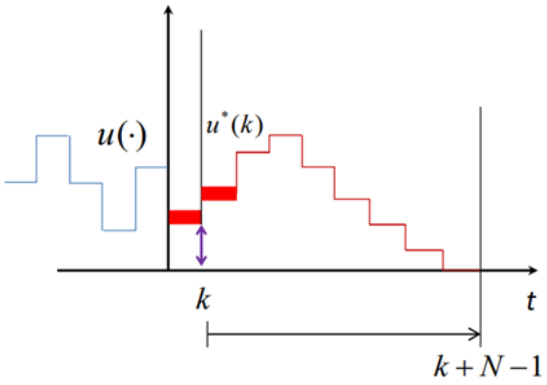
**Running Cost:**

Also known as stage cost:

$$l(x,u) = ||x_u - x^r||_Q^2 + ||u - u^r||_R^2 \tag{3.23}$$




(a) Running cost of state

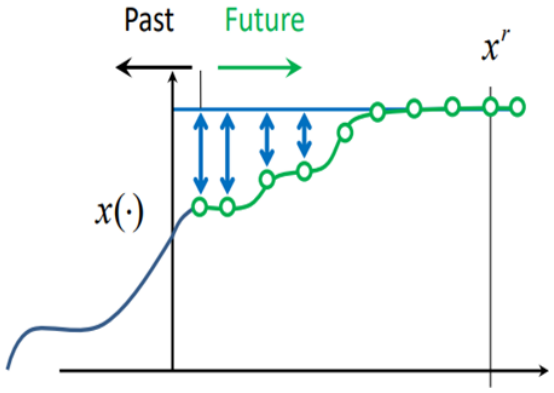



(b) Running cost of Input

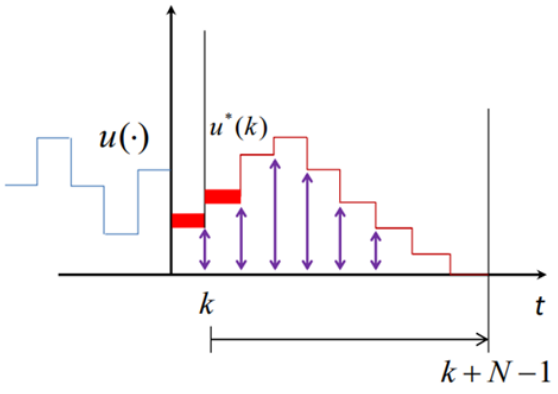
**Cost Function:**

Evaluation of the running costs along the whole prediction horizon.

$$J_N(x,u) = \sum_{(k=0)}^{N-1} l(x_u(k), u(k) \tag{3.24}$$




(a) Cost Function of State




(b) Cost Function of Input

**Optimal Control Problem (OCP):**

To find a minimizing control sequence. Evaluation of the running costs along the whole prediction horizon.

$$\frac{minimize}{u} J_N(x_\circ) = \sum_{(k=0)}^{N-1} l(x_u(k), u(k)) \tag{3.25}$$

$$subjectto : x_u(k+1) = f(x_u(k), u(k)) \tag{3.26}$$

$$x_u(0) = x_0, \tag{3.27}$$

$$u(k)\epsilon U, \forall k\epsilon[0, N-1] \tag{3.28}$$

$$x_u(k)\epsilon X, \forall k\epsilon[0, N] \tag{3.29}$$

**Value Function:**

Minimum of the cost function

$$V_N(x) = \frac{min}{u} J_N(x_0, u) \tag{3.30}$$

**Nonlinear Programming Problem (NLP):**

A standard problem formulation in numerical optimization having the general form:

$$\frac{min}{w} \emptyset(w) \qquad ObjectiveFunction \tag{3.31}$$

$$s.t. g_1(w) \leq 0 \qquad InequalityConstraints \tag{3.32}$$

$$g_2(w) = 0 \qquad equalityConstraints \tag{3.33}$$

Many NLP optimization algorithms (packages):-

- Ipopt
- Fmincon

For our problem we have choosen IPOPT for solving the NLP problem.

### Interior Point Optimizer (IPOPT)

It is an open-source software package for solving large-scale nonlinear optimization problems. It is widely used in the optimization community and is particularly well-suited for solving nonlinear programming problems with a large number of variables and constraints. IPOPT uses an interior point algorithm for solving nonlinear programming problems. This algorithm works by minimizing a barrier function that is a combination of the objective function and the constraints of the optimization problem. It can efficiently handle problems with tens of thousands of variables and constraints. It is also highly customizable, allowing users to tailor the algorithm to their specific problem requirements. IPOPT can handle a wide range of nonlinear programming problems, including problems with nonlinear equality and inequality constraints, nonconvex objective functions, and integer variables. It also provides options for handling various types of constraints, such as bound constraints, linear constraints, and nonlinear constraints. IPOPT is written in C++ and can be called from various programming languages, including MATLAB, Python, etc.

### Converting Problem Set from OCP to NLP

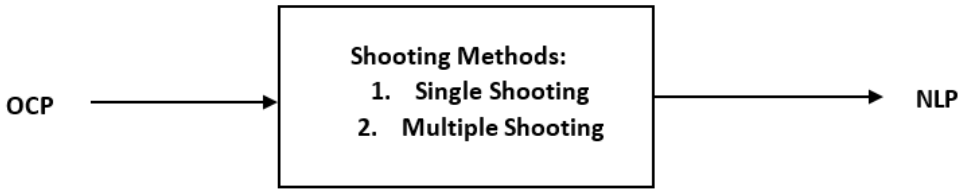



Figure 3.11: Single Step Action

### Single Shooting:

Single shooting is a numerical method used to solve optimal control problems (OCPs) that involve a finite number of decision variables and a set of dynamic constraints. It is a type of shooting method that involves dividing the time horizon of the problem into a finite number of subintervals, and solving the OCP by optimizing the control inputs over each subinterval separately. The basic idea behind single shooting is to first guess the values of the control inputs over each subinterval, and then integrate the system dynamics forward in time using these control inputs to obtain a predicted trajectory of the system. The predicted trajectory is then compared to the desired trajectory, and the control inputs are adjusted iteratively until the predicted and desired trajectories match.

Single shooting is computationally efficient and can handle problems with a large number of subintervals. However, it may not converge to the global optimum of the OCP and may require multiple initial guesses to find a good solution. Additionally, it may not be suitable for problems with complex dynamics or nonlinear constraints.
MAIN Drawback: Nonlinearity propagation: integrator function tends to become highly nonlinear for large N. Not a suitable method for nonlinear and/or unstable systems when optimizing over a long prediction horizon.

**Multiple Shooting:**

Multiple shooting is a numerical method used to solve optimal control problems (OCPs) that involve a finite number of decision variables and a set of dynamic constraints. It is a type of shooting method that involves dividing the time horizon of the problem into a finite number of subintervals, and solving the OCP by optimizing the control inputs and the state variables over each subinterval separately. The basic idea behind multiple shooting is to first guess the values of the control inputs and the state variables over each subinterval, and then use these guesses to solve a set of algebraic equations that relate the state and control variables at the end of each subinterval. These algebraic equations are known as continuity constraints, and they ensure that the state variables are continuous across the subintervals.

Multiple shooting is computationally efficient and can handle problems with a large number of subintervals. It also has better convergence properties compared to single shooting, as it ensures that the state variables are continuous across the subintervals. However, it may not converge to the global optimum of the OCP and may require multiple initial guesses to find a good solution. Additionally, it may not be suitable for problems with complex dynamics or nonlinear constraints.

**CASADI:**

CasADi is a software framework for numerical optimization and algorithmic differentiation, which could be used for solving optimization problems and developing optimal control solutions. It provides a powerful and efficient platform for modeling and solving complex optimization problems, with a wide range of features that include automatic differentiation, nonlinear programming, and real-time optimization. It provides efficient and accurate automatic differentiation tools, which allow users to compute gradients and Hessians of nonlinear functions without the need for symbolic differentiation. Also, it includes a suite of nonlinear programming solvers, which can be used to solve optimization problems with constraints and nonlinear objective functions. It has provided the interface on multiple platforms/interface

Python, Matlab which makes easy on scientific computation. The main advantage of it is Free and Open-Source, flexible and efficient, which makes it a powerful tool for solving complex optimization problems in a wide range of applications. CasADi can handle 4 standard problems.

- QP's (Quadratic programs)
- NLP's (Nonlinear programs)
- Root finding problems
- Initial-value problems in Ordinary Differential Equations (ODE)/ Differential Algebraic Equations (DAE)

# 4. Methodology

## 4.1 Flight Computer Working

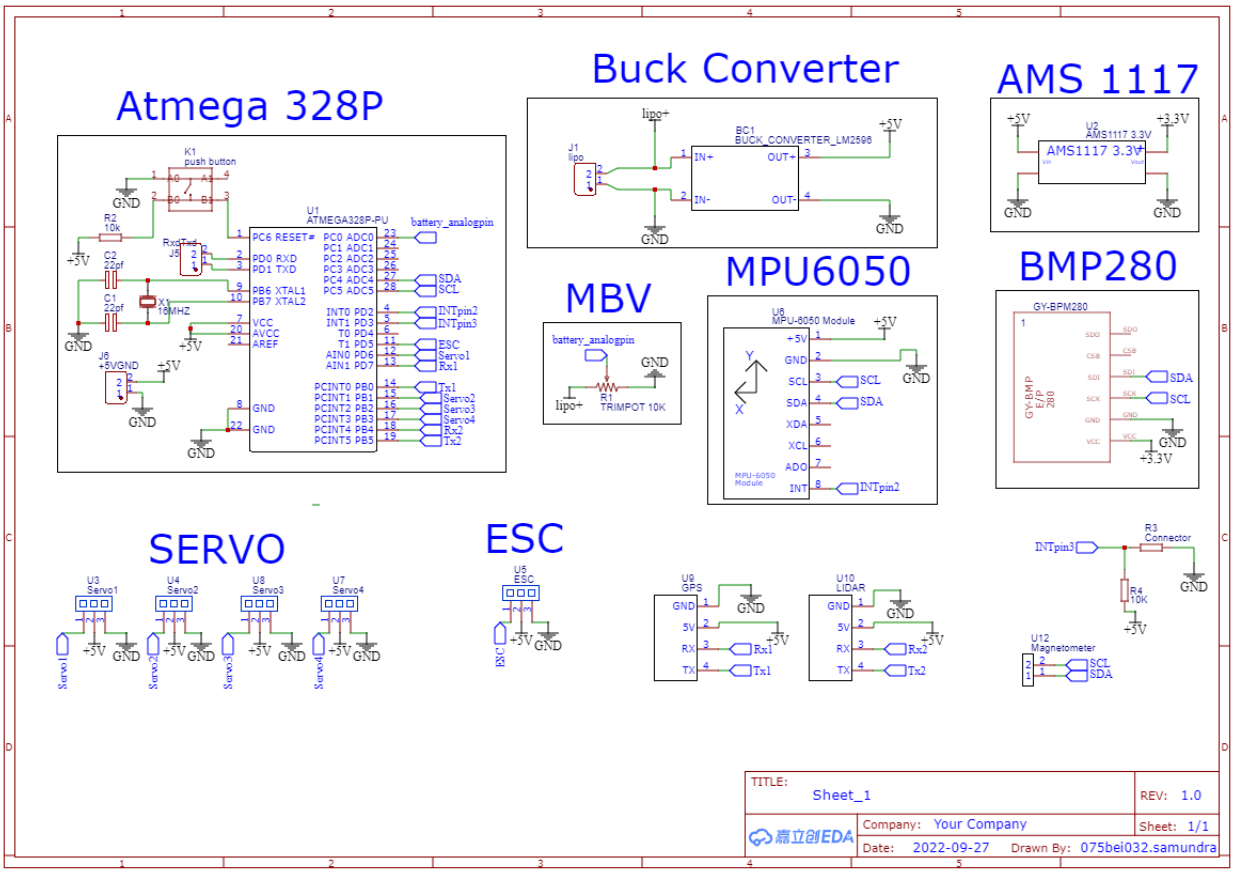



Figure 4.1: Atmega328P schematic diagram

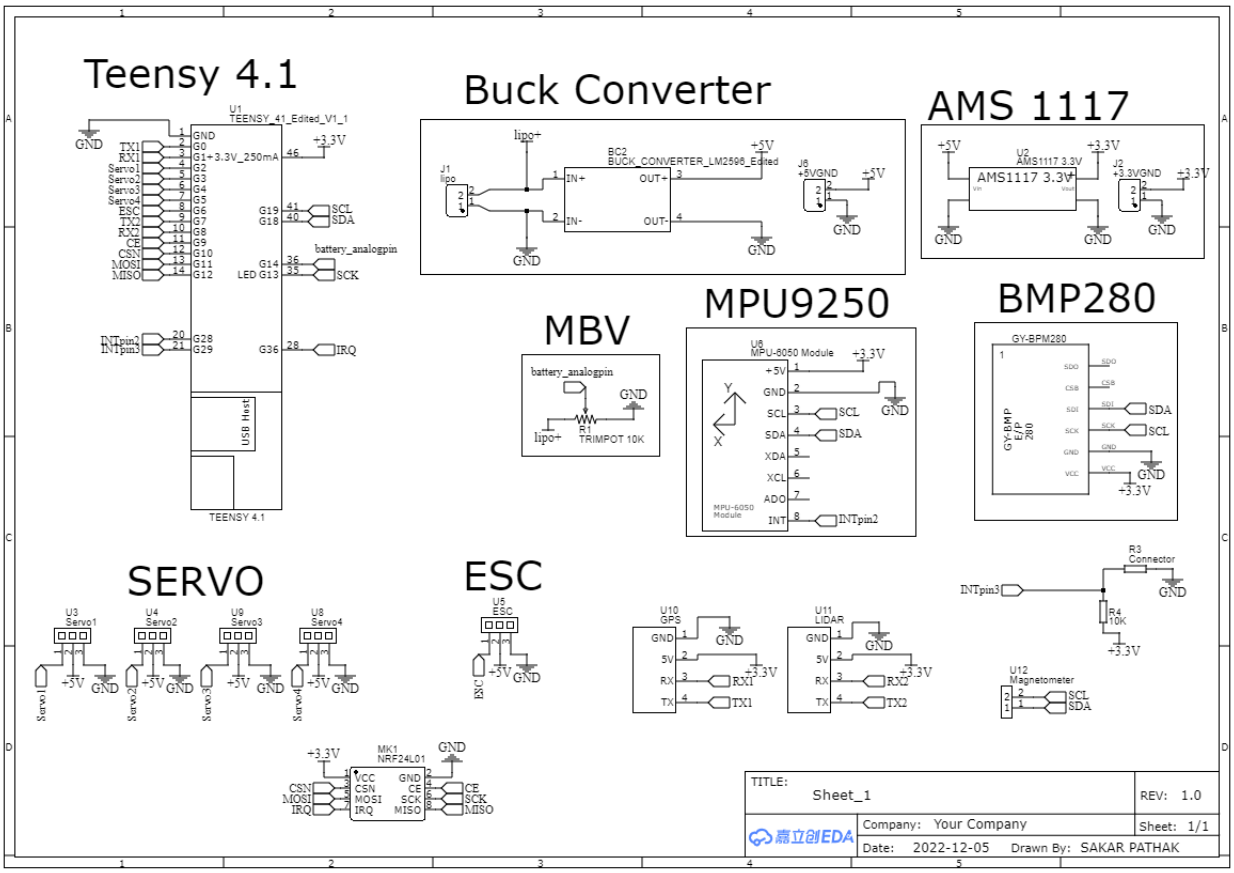



Figure 4.2: Teensy 4.1 schematic diagram

The LM2596 buck converter is a type of voltage regulator that is used to convert a higher voltage input to a lower voltage output. In this particular case, it is used to convert the input voltage from a 6S 65C 5000mAh lithium polymer battery (which typically provides a voltage output of around 22 volts) to 5 volts, which is then used to power the actuators in the system.

The 5V output from the buck converter is further regulated to 3.3V using the AMS1117 voltage regulator. The 3.3V output is then used to power the teensy microcontroller, NRF24L01, MPU9250, BMP280, lidar and GPS sensor. These components require a lower voltage input and hence, the voltage is further reduced using the voltage regulator.

Overall, the combination of the LM2596 buck converter and the AMS1117 voltage regulator provides a stable and regulated power supply to all the electronic components in the system, ensuring optimal performance and reliability.

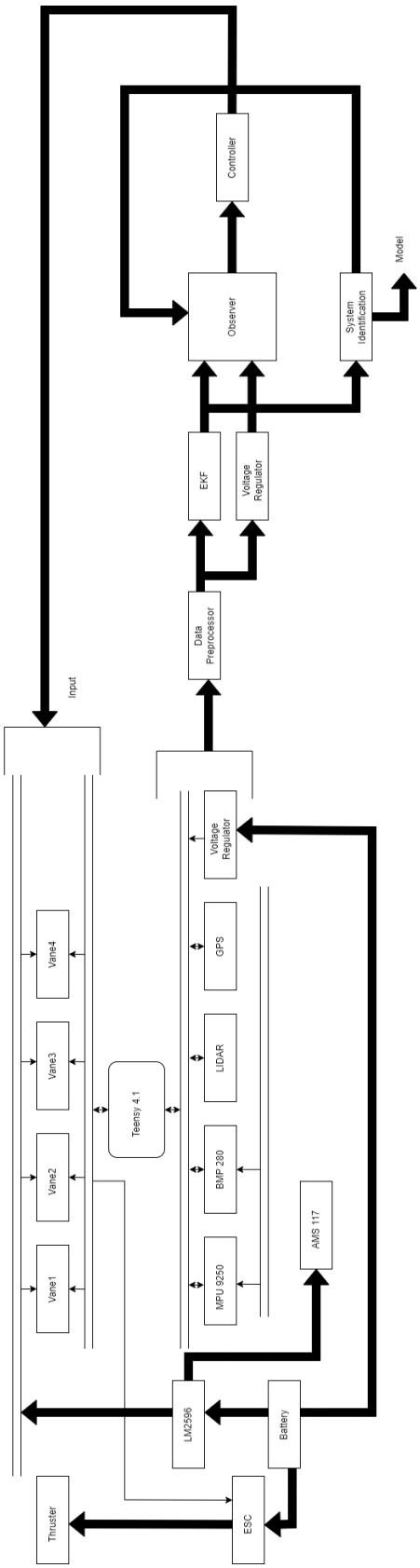



Figure 4.3: Project Architecture

Initially, an MPU6050 sensor, which had an accelerometer and a gyroscope, was used. However, it was found that the system's yaw angle was not satisfactory. Therefore, an upgrade was made to an MPU9150 sensor, which had an additional magnetometer, resulting in improved accuracy. To further improve accuracy, an MPU9250 sensor was utilized, which had additional sensors to enhance performance. In addition, a GPS module was incorporated to measure location accurately, which helped track the vehicle's movement concerning the desired trajectory. A BMP280 sensor was also utilized to measure altitude. However, it was noticed that inaccurate data was obtained in direct sunlight. Therefore, the BMP280 sensor was enclosed in a box that allowed air to pass through but blocked sunlight, resulting in more accurate altitude measurements. These sensor upgrades and improvements helped achieve a highly accurate and reliable vehicle tracking system. The methodology involved evaluating different sensors' performance, testing them in various scenarios, and analyzing the collected data to select the most appropriate sensors and improve the system's overall accuracy.

We also aimed to improve the accuracy of a vehicle tracking system by analyzing the impact of external factors on sensor performance. We observed that the magnetometer sensor worked fine when the EDF was not turned on, but it produced unreliable data when the vehicle was launched. Further analysis revealed that the sensor's position between the battery and EDF was the primary reason behind this issue. Due to high current passing through the wire from the battery, a magnetic field was generated, which ultimately affected the magnetometer's readings, leading to unreliable data. To address the issue of altitude measurement, we used a BMP280 sensor, which produced accurate results. However, for a soft touchdown, the altitude measurement needed to be very precise as the vehicle approached the ground. Therefore, an extra LIDAR sensor, with a range of up to 12m, was added to achieve more precise altitude measurements. For altitudes above 12m, the BMP280 sensor was used. The methodology involved analyzing sensor performance in different scenarios, identifying external factors affecting sensor accuracy, and selecting appropriate solutions to enhance the system's accuracy and reliability.

## 4.2 Thrust Vectoring mechanism (TVC)

We used two different approach in a progressive manner while coming in this phase of our vehicle model. Firstly, we used nozzle mechanism for controlling the thrust direction and controlling our VTOL and later proceed to using vanes as our thrust vectoring measure.

Unlike thrust vectoring nozzles, which can only provide limited control over the vehicle's direction of travel, vanes can actively manipulate the airflow around the vehicle to achieve precise control over its yaw.

Figure 4.4: Contrasting Thrust Vectoring Mechanism (TVC). First image is the use of nozzle while second is of use of vanes for TVC

By using vanes, the vehicle's flight can be controlled with greater accuracy, allowing for smoother transitions between hovering and forward flight modes. Additionally, vanes allow for better maneuverability in the air, which can be important in situations where the vehicle needs to navigate around obstacles or avoid collisions. A new flow control concept has been proposed that uses vanes to reduce inlet flow distortion and improve hover performance.

## 4.3 Thrust Test

To apply the controller, it is necessary to know the inputs given to the plant. In our case the actual input to the plant(vehicle) is the thrust generated by the motor. But we do not have direct control over the thrust generated by the motor. However it can be controlled through the throttle input which is actually a pwm signal given to the esc. At constant throttle the thrust of the motor changes with the change of battery voltage. So indirectly two parameters, throttle and battery voltage are affecting the thrust.

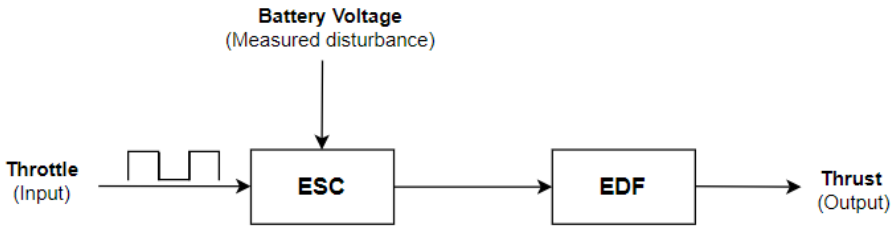



Figure 4.5: Block Diagram of Vehicle Input

### 4.3.1 Thrust Testing Bed

The motor was rigidly attached to the testing bed. A 10 kg load cell was used and the thrust data, battery voltage data, throttle data and time data were collected.

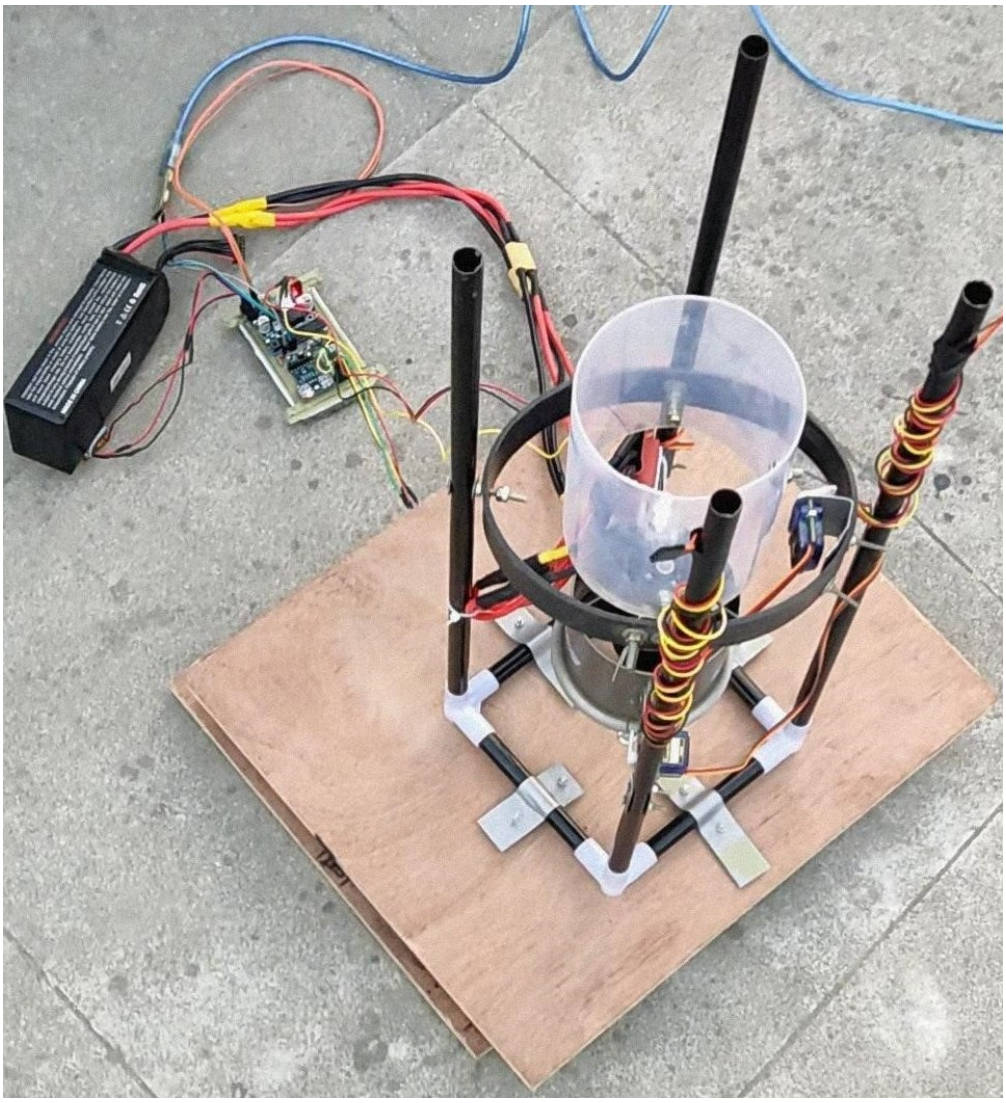


Figure 4.6: Thrust Testing Bed

## 4.4 Sensor Fusion

### 4.4.1 Preliminary Stage

**Setup:**

We acquired the necessary equipment, including the MPU 6050 sensor module, ATmega328P microcontroller, and a computer. We connected the MPU 6050 to the ATmega328P microcontroller and interfaced it with the computer.

**Sensor Calibration:**

We calibrated the sensors by determining the sensor bias and sensitivity to ensure accurate readings. The calibration procedure was performed by following the manufacturer's guidelines.

**Sensor Fusion Algorithm:**

We implemented a quaternion-based orientation estimation algorithm using the DMP of the MPU 6050 and ATmega328P microcontroller. The algorithm was designed to use the data from both the accelerometer and gyroscope sensors to estimate the orientation of the sensor in 3D space. We set the sample rate to 50 Hz.

**DataAcquisition:**

We acquired data by reading the quaternion data output by the DMP of the MPU 6050. The data was stored in a file format that was compatible with the software used for processing.

**Data Processing and Analysis:**

We processed the acquired data using the software. The quaternion data was converted to Euler angles. We analyzed the data to determine the accuracy and reliability of the sensor fusion algorithm. We evaluated the performance of the algorithm based on the accuracy of the orientation estimates.

### 4.4.2 Improved Stage:

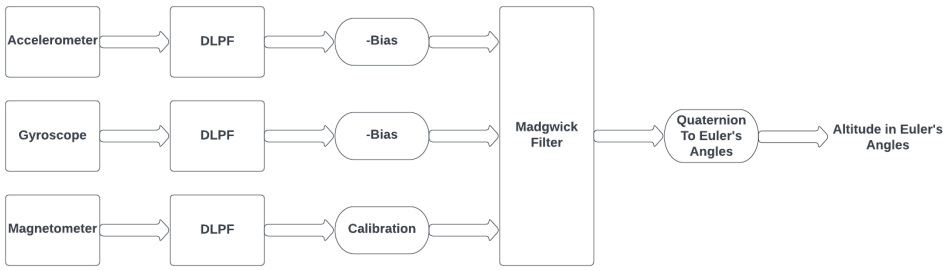



Figure 4.7: Sensor Fusion Block Diagram

**Setup:**

For this study, we used a different setup than the one described previously. We acquired a MPU9250 sensor module with a magnetometer, an ublox GPS module, a BMP280 altitude

sensor, and a Teensy 4.0 microcontroller. We connected the MPU9250, magnetometer, ublox GPS, and BMP280 to the Teensy 4.0 microcontroller and interfaced it with a computer.

**Sensor Calibration:**

We calibrated the MPU9250, magnetometer, and BMP280 sensors by determining the sensor bias and sensitivity to ensure accurate readings. The calibration procedure was performed by following the manufacturer's guidelines. The ublox GPS module was configured using the manufacturer's recommended settings.

**Calculation of Bias and Variance:**

To use raw data from MPU9250 and u-blox GPS for EKF parameters, we collected data and calculated mean, variance, and bias for each sensor measurement. Mean and variance were used as parameters in the EKF to provide accurate estimates of the system state, while bias accounted for systematic errors in sensor measurements to improve accuracy over time. Accurately characterizing sensor measurements and accounting for bias are important steps in preparing data for use in navigation and control applications with an EKF.

**Sensor Fusion Algorithm:**

We implemented a Madgwick filter for orientation estimation instead of the DMP used in the previous setup. The Madgwick filter is a gradient-based algorithm that uses the data from the MPU9250 sensors to estimate the orientation of the sensor in 3D space. We set the sample rate to 250 Hz. We also used an Extended Kalman filter to fuse the data from the ublox GPS and BMP280 sensors to estimate the position and altitude of the vehicle.

**Data Acquisition:**

We acquired data by reading the output of the Madgwick filter, Extended Kalman filter, and one-dimensional LIDAR. The data was stored in a file format that was compatible with the software used for processing.

**Data Processing and Analysis:**

We processed the acquired data using the software. The orientation, position, and altitude data were analyzed to determine the accuracy and reliability of the sensor fusion algorithm. We evaluated the performance of the algorithm based on the accuracy of the estimates.

For the altitude control, first the state space matrices of the rocket were derived and then the model was simulated in simulink for the trajectory of our need.

## 4.5 System Modeling

The methodology used involved the modeling of the VTOL system using Newton's law in the roll, pitch, and altitude parts. However, it was noted that there was no yaw control mechanism in the VTOL system at this stage, which meant that yaw was not included in the initial system modeling.

To address this limitation, a yaw control mechanism was added to the VTOL system, and the new system was analyzed. With the addition of the yaw control mechanism, the system was now complete and able to operate in all four dimensions: yaw, pitch, roll, and altitude.

After the completion of the system modeling, system identification was carried out in all four dimensions of the VTOL system: yaw, pitch, roll, and altitude. This involved the use of data collected from various sensors on the VTOL system, which was then analyzed using mathematical models to identify the behavior and dynamics of the system.

### 4.5.1 State Space Model for Altitude Control from First Principle

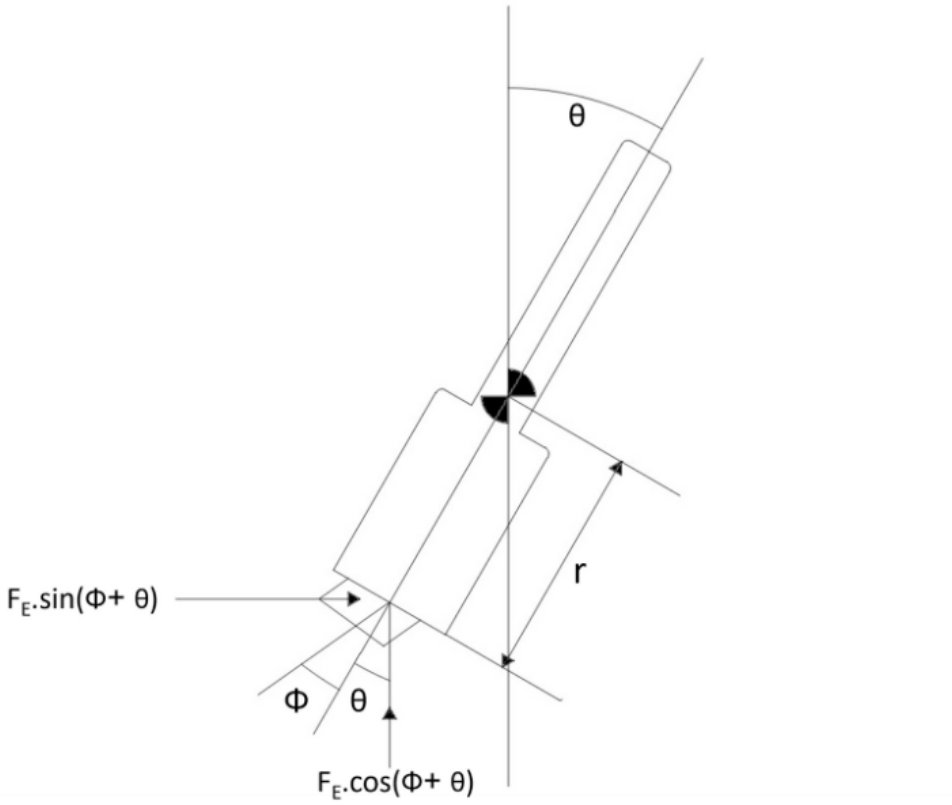



Figure 4.8: Rotational Dynamics of the Vehicle

$\theta$ = angle made by the vehicle with earth vertical -axis in degrees
$\phi$ = angle made by the nozzle with vehicle's vertical axis in degrees
$F_E$ = force generated by the motor in Newton
R = distance from center of mass to the nozzle in meters
It is assumed that the thrust is generated at the top end of the nozzle.
Force exerted in the direction of earth horizontal axis = $F_E sin(\phi + \theta)$
Force exerted in the direction of earth vertical axis = $F_E cos(\phi + \theta)$
$a_g$ = acceleration due to gravity = $9.81 m/s^2$
z = altitude in meters
$v_z$ = velocity in z direction in meters per second
$a_z$ = net acceleration in z direction in meters per $second^2$
Sampling Time ($T_s$) = 20 milliseconds
Mass of rocket (m) = 2.5kg

we have,

$$a_{z(k)} = \frac{F_{E(k)}.cos(\phi_{(k)} + \theta_{(k)})}{m} - a_g \tag{4.1}$$

From equation (7) $a_{z(k)}$ is maximum when $F_{E(k)}$ is maximum and $a_{z(k)}$is minimum when $F_{E(k)}$ is minimum.

$$0 \leq F_{E(k)} \leq 35.28N \tag{4.2}$$

$$-9.81m/s^2 \leq a_{z(k)} \leq 4.302m/s^2 \tag{4.3}$$

$$z_{(k+1)} = z_{(k)} + T_s.v_{z(k)} + \frac{T_s^2.a_{z(k)}}{2} \tag{4.4}$$

$$v_{z(k+1)} = v_{z(k)} + T_s.a_{z(k)} \tag{4.5}$$

Now converting equation (10) and (11) into state space form taking $a_{z(k)}$ as input. $a_{z(k)}$ can be converted to throttle using equation (7) and (6).

$$\begin{bmatrix} z_{(k+1)} \\ v_{z(k+1)} \end{bmatrix} = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} . \begin{bmatrix} z_(k) \\ v_{z(k)} \end{bmatrix} + \begin{bmatrix} \frac{T_s^2}{2} \\ T_s \end{bmatrix} . \begin{bmatrix} a_{z(k)} \end{bmatrix} \tag{4.6}$$

$$\begin{bmatrix} z_{(k)} \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix} . \begin{bmatrix} z_{(k)} \\ z_{(k)} \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} . \begin{bmatrix} a_{z(k)} \end{bmatrix} \tag{4.7}$$

The controllability of the rocket was verified in matlab.

Here acceleration in z-direction is input given to the system while altitude is the system's output. For a real plant the acceleration in z-direction can be converted to the throttle using equations (7) and (6). A predefined input to the vehicle was simulated which gave altitude response of our need.

### 4.5.2 State Space Model for Pitch, Roll Control from First Principle

For attitude control of the vehicle moment of inertia (I) and distance between center of mass and nozzle is required.

**Determination of I and r**

The moment of inertia ($I$) of the vehicle was determined experimentally. Two strings of fixed length were fixed at either side of the center of mass of the vehicle at an arbitrarily fixed distance. Then the vehicle was oscillated about the center of mass. The time period of oscillation was recorded. Distance between center of mass and TVC nozzle was measured. The mass of the vehicle was also measured.

Following data were obtained from the experiment:
Mass($m$) = 2.5 kg
Distance between center of mass and the string attached point($d$) = 0.3 m
String Length ($l$) = 0.205m
Distance between center of mass and the nozzle ($r$) = 0.46m

$$I = \frac{mgt^2d^2}{4\pi^2 l} = 0.2298 kgm^2 \tag{4.8}$$

where,
$g$(acceleration due to gravity) = 9.81 $m/s^2$
$t$(average time period of single oscillation) = 0.918 seconds

Therefore $I$ was determined to be 0.2298 $kgm^2$ and $r$ was determined to be 0.46$m$.

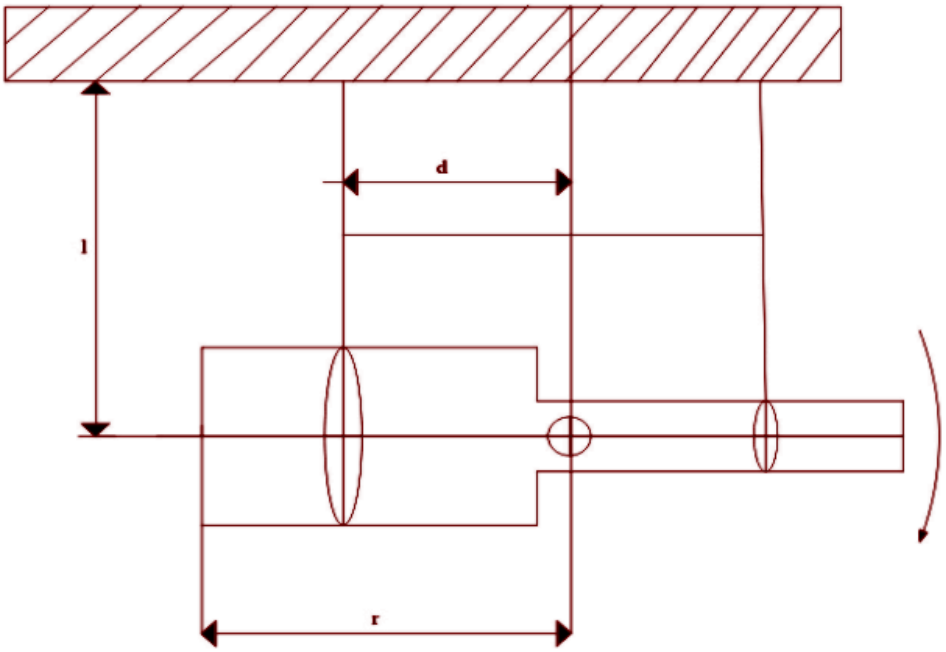



Figure 4.9: Determination of $I$ and $r$

**Pitch, Roll Control**

Due to the absence of a yaw control mechanism the yaw of the vehicle is not controllable. But the pitch and roll can be controlled by appropriately vectoring the nozzle with the two servo motors attached with it.

From figure (13)

$\omega$ = angular velocity in $degrees/sec^2$

$\alpha$ = angular acceleration in $degrees/sec^2$

$I$ = moment of inertia in $kgm^2$

$r$ = distance between center of mass and the nozzle

$$I\alpha = -F_E sin\phi.r$$

$$\alpha = -\frac{F_E sin\phi.r}{I} \tag{4.9}$$

$$\theta_{(k+1)} = \theta_{(k)} + T_s.\omega_{(k)} + \frac{T_s^2.\alpha_{(k)}}{2} \tag{4.10}$$

$$\omega_{(k+1)} = \omega_{(k)} + T_s.\alpha_{(k)} \tag{4.11}$$

From (15) for small $\phi_{(k)}$

$$\alpha_{(k)} = -\frac{F_{E(k)}\phi_{(k)}.r}{I} \tag{4.12}$$

From equations (16), (17) and (18)

$$\theta_{(k+1)} = \theta_{(k)} + T_s.\omega_{(k)} - \frac{T_s^2.r(F_{E(k)}.\phi_{(k)})}{2I} \tag{4.13}$$

$$\omega_{(k+1)} = \omega_{(k)} - \frac{I_s.r(F_{E(k)}.\phi_{(k)})}{I} \tag{4.14}$$

Converting equations (19) and (20) to state space form

$$\begin{bmatrix}\theta_{(k+1)}\\ \omega_{(k+1)}\end{bmatrix} = \begin{bmatrix}1 & T_s\\ 0 & 1\end{bmatrix}.\begin{bmatrix}\theta_{(k)}\\ \omega_{(k)}\end{bmatrix} + \begin{bmatrix}\frac{-T_s^2.r}{2I}\\ \frac{-T_s.r}{I}\end{bmatrix}.\begin{bmatrix}F_{E(k)}.\phi_{(k)}\end{bmatrix} \tag{4.15}$$

$$\begin{bmatrix}\theta_{(k)}\\ \omega_(k)\end{bmatrix} = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}.\begin{bmatrix}\theta_{(k)}\\ \omega_{(k)}\end{bmatrix} + \begin{bmatrix}0\\ 0\end{bmatrix}.\begin{bmatrix}F_{E(k)}.\phi_{(k)}\end{bmatrix} \tag{4.16}$$

$$x_{(k+1)} = Ax_{(k)} + Bu_{(k)} \tag{4.17}$$

$$y_{(k)} = Cx_{(k)} + Du_{(k)} \tag{4.18}$$

where,

$$x_{(k)} = \begin{bmatrix}\theta_{(k)}\\ \omega_{(k)}\end{bmatrix} \tag{4.19}$$

$$y_{(k)} = \begin{bmatrix}\theta_{(k)}\\ \omega_{(k)}\end{bmatrix} \tag{4.20}$$

Hence,

$$A = \begin{bmatrix}1 & T_s\\ 0 & 1\end{bmatrix} \tag{4.21}$$

$$B = \begin{bmatrix}-T_s^2/2I\\ -T_s.r/I\end{bmatrix} \tag{4.22}$$

$$C = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} \tag{4.23}$$

$$D = \begin{bmatrix}0\\ 0\end{bmatrix} \tag{4.24}$$

From the above state space equations the input is $F_{Ek}.\phi_k$. The nozzle angle $\phi_k$ can be calculated from the input by dividing with thrust. The thrust input is given by altitude controller from equation (7).

$$\phi_{(k)} = u_{(k)}/F_{E(k)} \tag{4.25}$$

### 4.5.3 Block diagram of system for describing the movement around yaw,pitch and roll axis

**insert image**

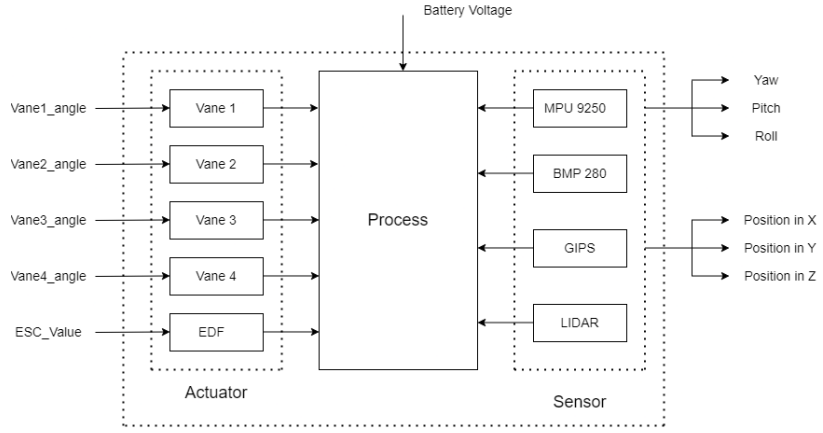



Figure 4.10: Abstract Block Diagram of System for Describing the Movement Around YPR Axis

In order to make our system more easy to model and control, we decided to perform system identification separately for yaw, pitch, roll, and altitude. For the control of yaw, pitch, and roll, we have five control inputs in total, which are the duty cycle of the PWM signals to the four servo motors that control the vanes, and the ESC connected to EDF motor which acts as a thruster. However, in order to capture the dynamics of the system more accurately, we considered the battery voltage as a measured disturbance to the system. The outputs measured were the angles in yaw, pitch, and roll axes. By identifying the system separately for each axis, we were able to obtain more accurate models for each specific control objective, which would enable us to design better controllers for our system.

### 4.5.4 Time Variance of Battery Voltage

In our system, the time invariance property is an important consideration for accurate modeling and control. However, we observed that the battery voltage is not time-invariant and changes over time due to discharging. This means that the thrust of the motor at the

same input PWM value also changes with time, leading to a violation of the time invariance property of the system.

To address this issue, we incorporated the battery voltage as a measured disturbance in our system identification process. By doing so, we were able to account for the varying thrust of the motor due to the changing battery voltage and model the system more accurately. With the battery voltage as a measured disturbance, we were able to approximate the behavior of our system as a linear time-invariant system. This allowed us to use standard control techniques such as LQR and MPC for control design and optimization. By accounting for the non-time-invariant battery voltage in our system model, we were able to achieve better control performance and ensure that our system operates reliably over time.

### 4.5.5 ARX method for estimating the transfer function for yaw movement of the rocket

To estimate the transfer function for the yaw control system, we used the ARX model. In our case, we used the ARX model to predict the output yaw angle for a given vanes angle.

However, the ARX model has some limitations. It can only handle SISO (Single Input Single Output) systems, so we had to neglect the effect of the ESC value in our model. Furthermore, the measured input disturbance, which was the battery voltage in our case, had to be neglected as well.

Despite these limitations, the ARX model provided us with a useful transfer function that we could use for controller design and trajectory tracking.

### 4.5.6 ARMAX method for estimating the transfer function for yaw movement of the rocket

After observing the auto-correlation of the residuals from the ARX model, we found that the residuals were correlated up to two error terms behind, indicating the presence of a second-order moving average (MA) process. Thus, an ARMAX model was fitted with the order parameters [2 1 8 2], where the additional parameter 2 represents the order of the MA process. By including the additional MA term, we aimed to capture the remaining unexplained dynamics in the system and improve the accuracy of the model.

### 4.5.7 Subspace method for estimating the state space for yaw movement of the rocket

We had two inputs that could directly change the yaw angle of the rocket: the vanes angle and the thruster. While the vanes angle could both increase or decrease the yaw angle of the

rocket, the thruster could only decrease the angle, rotating the rocket in only a clockwise direction. We considered the battery voltage as the measured disturbance. To identify the whole system, we followed the following steps. First, we set the equilibrium value for the thruster's thrust equal to the rocket's mass. Then, we found the corresponding ESC value that would make the motor generate this amount of thrust at peak battery voltage. We took this ESC value as the equilibrium value for the input ESC. Similarly, we found the equilibrium value for the vanes angle by giving the motor the equilibrium thrust and selecting the vanes angle that could almost cancel out the clockwise rotation of the rocket due to the equilibrium thrust. After finding the equilibrium values for both inputs, we kept one input constant at the equilibrium position and changed the other between two equidistant values from the equilibrium position. Then, we conducted experiments and obtained data. To better model the system, we preprocessed the data. We removed the initial bias from the yaw and yaw velocity data, as well as from the ESC value and battery voltage. We also noticed that the battery voltage got subjected to a large amount of noise due to the use of buck converters which use switching in high frequency and edf which uses a large amount

**Note:** During the system identification process we kept the battery voltage as a second input. Later after the A B C D and K matrix are computed the last column of the B matrix is substituted to the D matrix and the second input is removed from the input matrix and placed on the disturbance matrix.

### Keeping ESC value in equilibrium

we initially used a pseudo-random binary sequence (PRBS) signal for the input vanes angle. However, the results obtained with the PRBS signal were very poor. The system took a very long time to show all its dynamics with the PRBS signal, but the battery could only last for less than 2 minutes. As a result, we had to abandon the idea of using PRBS and instead changed the vanes angle between two values based on the angular rate getting increased or decreased by two values with the same magnitude but opposite direction. This method proved to be much more effective in terms of capturing the system dynamics within the battery life constraints.

### Keeping vanes angle value in equilibrium

In contrast to the previous case, in this scenario, we used a pseudo-random binary sequence (PRBS) signal to excite the input thruster. The system was run for a duration of 14 seconds, during which all of the system's important dynamics were captured. The use of a PRBS signal proved to be effective in exciting the system's dynamics and capturing them within

a relatively short amount of time. This approach allowed us to obtain comprehensive data for system identification and model development. Overall, this experience highlights the importance of carefully selecting the appropriate input signal for system identification and model development. By utilizing a PRBS signal in this case, we were able to efficiently capture the system dynamics and design an effective control system.

## 4.6 Control

### 4.6.1 PID Tuning

Tuning a PID controller is the process of adjusting its parameters to achieve the desired control response. PID tuning methods: Manual Tuning:

**Trial and error method:**

This method involves manually adjusting the values of Kp, Ki, and Kd until the desired control performance is achieved. This method is simple but can be time-consuming and may not guarantee optimal performance.

**Ziegler-Nichols method:**

- Start with proportional gain (Kp):- Set the integral and derivative gains to zero and increase the proportional gain until the system starts to oscillate. This is known as the critical gain or ultimate gain (Ku).
- Determine the ultimate period (Tu): Measure the time it takes for the system to complete one full oscillation cycle at the critical gain. This is the ultimate period.
- Calculate the controller gains: According to Ziegler-Nichols method to calculate the proportional, integral, and derivative gains based on the critical gain and ultimate period. The formulas for the gains are as follows:
  - Kp = 0.6Ku
  - Ki = 1.2 $\frac{Ku}{Tu}$
  - Kd = 0.075 Ku Tu
- Adjust the gains: Once you have calculated the gains, adjust them as necessary to achieve the desired control response. Increasing the proportional gain will increase the system's response time, while increasing the integral gain will reduce steady-state error. Increasing the derivative gain will improve the system's stability.

- Test and repeat: Test the system's response to the adjusted gains and repeat the tuning process as necessary until the desired control response is achieved.

### 4.6.2 Automatic Tuning:

It use algorithms and software to adjust the PID gains based on system response data, without requiring manual intervention.

**Matlab Simulink:**

There are several built-in tools that can be used to automatically tune PID controllers. One of the most commonly used tools is the PID Tuner, which is part of the Control System Toolbox. Here's how you can use the PID Tuner to automatically tune a PID controller:

- Add a PID Tuner block to the model along with PID controller.
- Open the PID Tuner block and select "Design Mode".
- In Design Mode, you can specify the desired response of the system using the built-in response optimization criteria. For example, you can specify a step response with a settling time of 0.1 seconds and a maximum overshoot of 10
- Click on "Tune" to automatically tune the PID gains based on the specified response criteria.
- The PID Tuner will generate a new set of PID gains that meet the specified response criteria, which you can apply to the PID controller in the Simulink model.
- Proportional Integral Derivative (PID) Controller

The PID Tuner also provides advanced tuning options, such as the ability to specify constraints on the PID gains, to add additional filters to the controller, and to specify different types of input signals for the system.

**sssamundra**

### 4.6.3 LQG Control

**State Estimation with Kalman Filters**

Kalman Filter is an optimal state estimator. In this project, we have employed kalman filters in the roll and pitch of the rocket to estimate angle and angular velocity. The following are the equations for kalman filter:

Prediction Phase

$$x_{(k)} = Ax_{(k-1)} + Bu_{(k)} \tag{4.26}$$

$$P_{(k)} = AP_{(k-1)}A^T + Q \tag{4.27}$$

Correction Phase:

$$K_G = \frac{P_{(k)}C^T}{CP_{(k)}C^T + R} \tag{4.28}$$

$$x_{(k)} = x_{(k)} + K_G(y_{(k)} - Cx_{(k)}) \tag{4.29}$$

$$P_{(k)} = (I + K_G C)P_{(k)} \tag{4.30}$$

Here, A,B and C matrices are the same as that of the state space equations in (27), (28) and (29). P is the associated uncertainty associated with state matrix x. Q and R are the covariance matrices.

**LQG with Setpoint tracking and integral action for pitch roll control:**

For integral action and setpoint tracking new matrices were made with some augmentations from the original matrices.

$$z_{(k+1)} = A_{aug}.z_{(k)} + B_{aug}.\Delta u_{(k)} \tag{4.31}$$

$$e_{(k)} = C_{aug}.z_{(k)} \tag{4.32}$$

where,

$$z_{(k)} = \begin{bmatrix} \Delta x_{(k)} \\ e_{(k)} \end{bmatrix} = \begin{bmatrix} \theta_{(k)} - \theta_{(k-1)} \\ \omega_{(k)} - \omega_{(k-1)} \\ r(\theta) - \theta_{(k)} \\ r(\omega) - \omega_{(k)} \end{bmatrix} \tag{4.33}$$

where,
r($\theta$) = set point for angle
r($\omega$) = set point for angular velocity

$$A_{aug} = \begin{bmatrix} A & 0 \\ -CA & I \end{bmatrix} \tag{4.34}$$

$$B_{aug} = \begin{bmatrix} B \\ -CB \end{bmatrix} \tag{4.35}$$

$$C_{aug} = \begin{bmatrix} 0 & I \end{bmatrix} \tag{4.36}$$

where, $A_{aug}$ , $B_{aug}$, $C_{aug}$ are matrices obtained by augmenting the A, B and C matrices respectively from (27), (28) and (29)
$\Delta u_k$ is the required change in input

Now the cost function for the plant is:

$$cost function : \sum {Z_{(k+i)}}^T \bar{Q} Z_{(k+i)} + \Delta {u_{(ki)}}^T . R . \Delta u_{(k+i)} \tag{4.37}$$

where,

$$\bar{Q} = \begin{bmatrix} 0 & 0 \\ 0 & Q \end{bmatrix} \tag{4.38}$$

we get,

$$cost function : \sum {e_{(k+i)}}^T Q e_{(k+i)} + \Delta {u_{(k+i)}}^T . R . \Delta u_{(k+i)} \tag{4.39}$$

From LQR,

$$\Delta u_{(k+1)} = -k z_{(k)} \tag{4.40}$$

where, $Q$ is the weight given to the states to the states and R is the weight given to the input.
This cost function is solved to get an optimal feedback matrix K using the Riccati equation in matlab.

**Implementation of LQG**

The LQG is implemented in the microcontroller as shown in the block diagram below. Here Θrepresents the estimated values from measured values and input using Kalman filter.

First, the sensors in the vehicle give the output $y_{(k)}$. The output is then passed through the kalman filter along with the actuator's input $u_{(k)}$ which then gives an estimate of the states of the vehicle as $\hat{x}_{(k)}$.
Now for set point tracking and integral action the estimated values are subtracted with the previously estimated values. A new column matrix $z_{(k)}$ is formed with the difference of estimates and error. Error is calculated by subtracting the estimates from the set points.

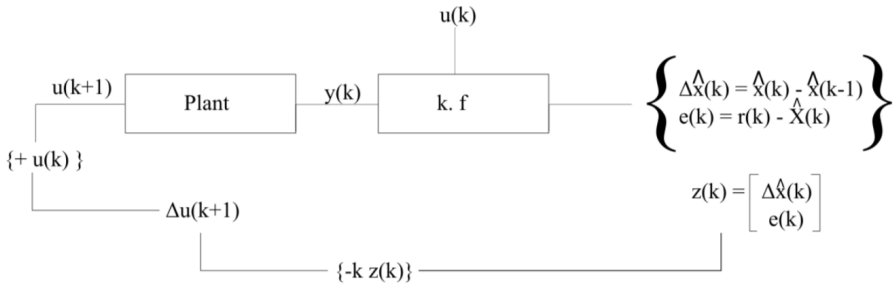



Figure 4.11: Implementation of LQG

The $z_{(k)}$ is then multiplied by the optimal gain matrix $K$ and negated to get the required change in input $\Delta u_{(k+1)}$ as,

$$\Delta u_{(k+1)} = -K \times z_{(k)} \tag{4.41}$$

Also,

$$\Delta u_{(k+1)} = u_{(k+1)} - u_{(k)} \tag{4.42}$$

Hence,

$$u_{(k+1)} = \Delta u_{(k+1)} + u_{(k)} \tag{4.43}$$

### 4.6.4 Flow Chart of MPC:

- System identification
  - Develop a mathematical model of the system
  - Identify the system parameters
- Prediction
  - Use the mathematical model to predict the behavior of the system over a finite time horizon
  - Take into account the current state of the system and the predicted future inputs
- Optimization
  - Formulate an optimization problem that minimizes a cost function subject to constraints
  - The cost function is typically a weighted combination of performance objectives such as tracking a reference trajectory, minimizing control effort, or satisfying constraints

- Solution
  - Solve the optimization problem using numerical optimization techniques such as quadratic programming or nonlinear programming
  - The solution provides the optimal inputs over the finite time horizon that minimize the cost function subject to the constraints
- Implementation
  - Apply the first control input from the optimal solution to the system
  - Re-compute the optimization problem at each time step using updated measurements of the system state and any new reference trajectory or constraint information
  - Repeat steps 2-5 at each time step

### 4.6.5 Single Shooting Algorithm

- Divide the time horizon into a finite number of subintervals, and discretize the control inputs over each subinterval.
- Guess the initial values of the control inputs over each subinterval.
- Integrate the system dynamics forward in time using these control inputs to obtain a predicted trajectory of the system.
- Compare the predicted trajectory to the desired trajectory, and calculate the error between the two trajectories.
- Use the error to update the values of the control inputs over each subinterval, and repeat steps 3-5 until the predicted and desired trajectories match.
- Once the optimal control inputs have been found, use them to integrate the system dynamics forward in time to obtain the optimal trajectory of the system.

### 4.6.6 Multiple Shooting Algorithm

- Divide the time horizon into a finite number of subintervals, and discretize the control inputs and the state variables over each subinterval.
- Guess the initial values of the control inputs and the state variables over each subinterval.

- Use the guessed values to integrate the system dynamics forward in time over each subinterval, and calculate the values of the state variables at the end of each subinterval.
- Use the calculated values of the state variables and the guessed values of the control inputs to solve a set of algebraic equations that relate the state and control variables at the end of each subinterval. These equations are known as continuity constraints.
- Use the continuity constraints to update the guessed values of the state variables and the control inputs over each subinterval, and repeat steps 3-5 until the predicted and desired trajectories match.
- Once the optimal control inputs and state variables have been found, use them to integrate the system dynamics forward in time to obtain the optimal trajectory of the system.

# 5. Results & Discussion

## 5.1 Thrust Test at Varying Throttle

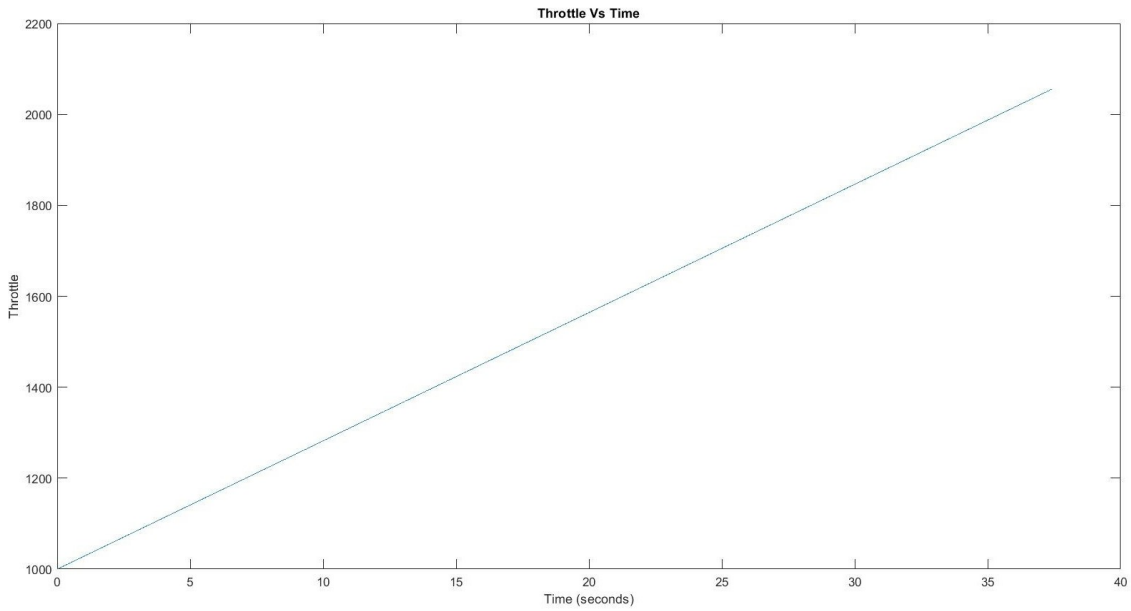



Figure 5.1: Throttle vs Time

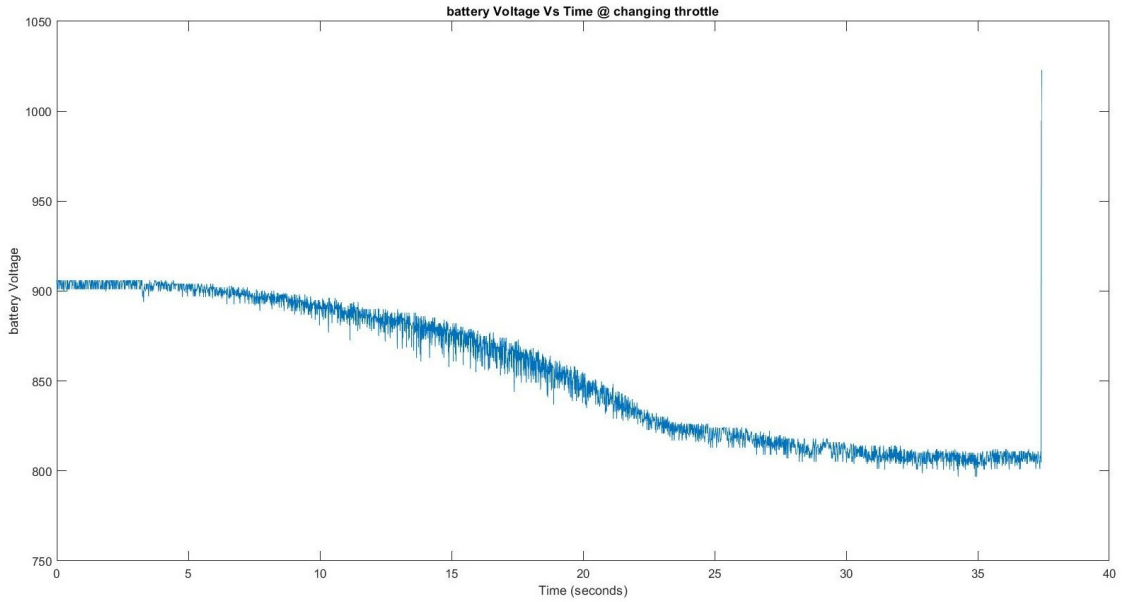



Figure 5.2: Battery Voltage vs Time at Changing Throttle

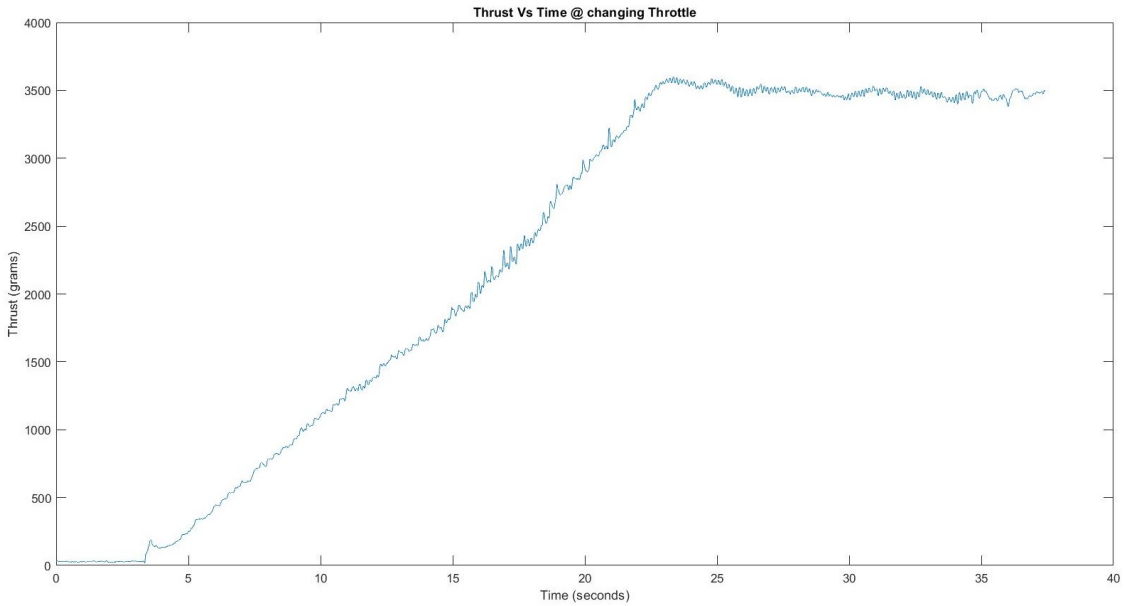



Figure 5.3: Thrust vs Time at Changing Throttle

As the throttle to the ESC was increased over time, the output thrust of the motor also increased. However, due to the slow decrement of the thrust from the EDF, the thrust output decreased as the battery voltage dropped over time. This data has proven to be very valuable in determining the appropriate throttle value for the rocket's weight, which is crucial during system identification and control.

## 5.2 Comparison Between MPU's internal DMP, Madgwick Filter and EKF

The digital motion processor (DMP) of the MPU6050 is a powerful tool for obtaining high-frequency data, with a theoretical limit of up to 1000Hz. However, during practical implementation, only 167 samples per second were achieved. This lower sampling rate could potentially affect the control of the rocket. To address this issue, a madgwick filter was used in combination with a teensy 4.1 microcontroller. This allowed for a sampling rate of more than 900Hz, but to prevent oversampling, the sampling rate was set to 250Hz. This resulted in a smooth running system. While an extended Kalman filter (EKF) was used to fuse data from the MPU and GPS for attitude and position estimation, it was found that the attitude estimates, particularly the yaw angle, suffered from noise and drift. However, the position estimates from the EKF were good. To address this issue, a combination of the madgwick filter and EKF was used. The madgwick filter was used for attitude estimation, while the EKF was used for position estimation. The beta parameter of the madgwick filter was ini-

tially set to a very small value, which caused a significant delay in the system. After tuning the parameter to 0.3, an optimal balance was achieved between delay and accuracy for our specific application. Overall, the combination of the madgwick filter and EKF allowed for accurate and reliable estimation of both attitude and position, despite the limitations of the DMP sampling rate.

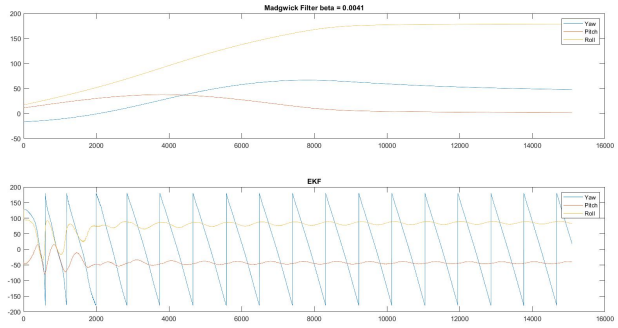



Figure 5.4: Madgwick vs EKF for Yaw, Pitch, Roll Stationary Data

| **Standard deviation** | **Yaw** | **Pitch** | **Roll** |
|---|---|---|---|
| Madgwick | 2.4291 | 0.4923 | 0.3536 |
| EKF | 105.6154 | 2.5318 | 3.1724 |

Table 5.1: Standard deviations for Madgwick and EKF

The standard deviation of the Madgwick filter can be even more increased by properly tuning the beta parameter.

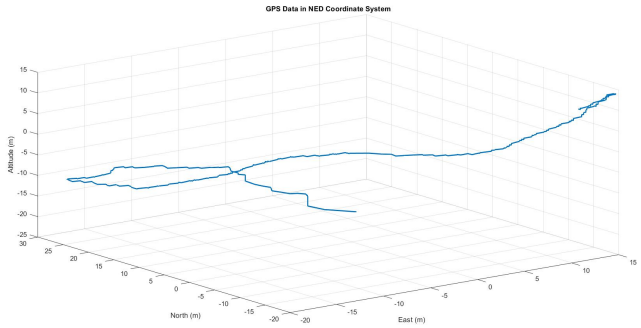



Figure 5.5: GPS Stationary Data in NED Coordinate System

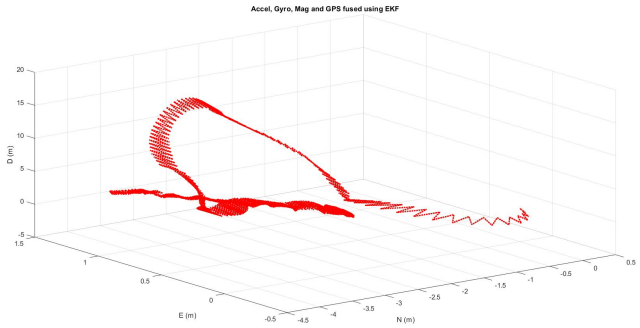



Figure 5.6: GPS Stationary Data After Sensor Fusion with IMU Data in NED Coordinate System

| **Standard deviation** | **North** | **East** | **Down** |
|---|---|---|---|
| GPS | 16.3388m | 11.7160m | 8.9393m |
| EKF | 0.5536m | 0.3961m | 4.1700m |

Table 5.2: Standard deviations for GPS and EKF

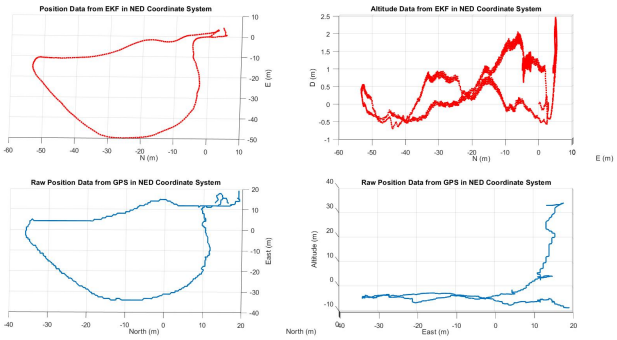



Figure 5.7: Comparison of EKF Output with GPS data

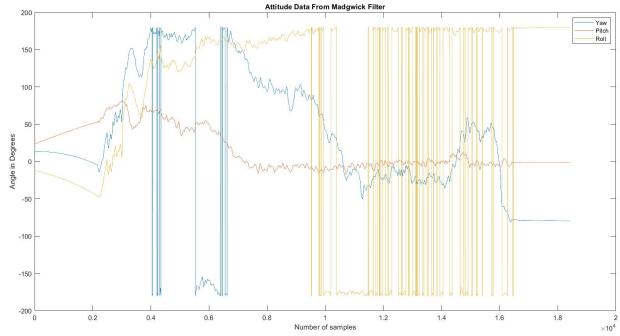



Figure 5.8: Attitude Data From Madgwick Filter

## 5.3 ARX Model

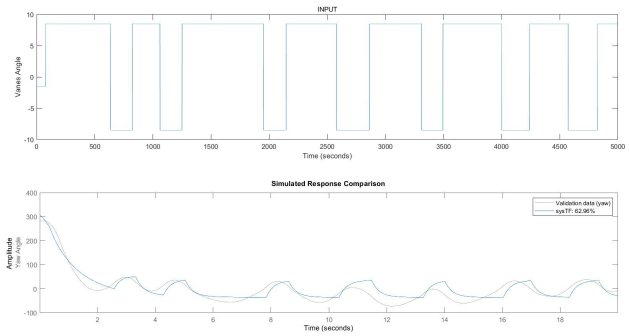



Figure 5.9: Input and Output Comparison Plot for ARX Model Estimation

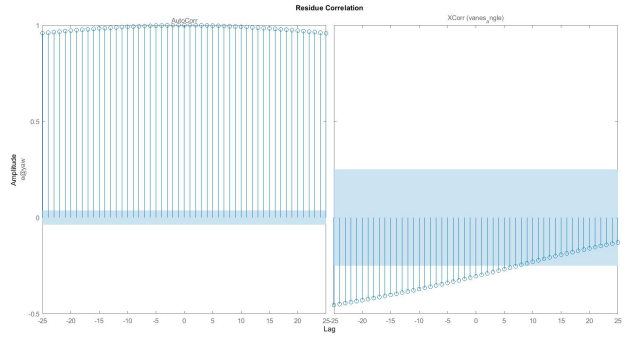



Figure 5.10: Error Correlation Plot for ARX Model Estimation

After obtaining the predicted data using the estimated transfer function, we calculated the autocorrelation and cross-correlation between the prediction error and the input data. We found that the correlation values were significant, indicating that there was still some information left in the prediction error that was not captured by the estimated transfer function. This means that the estimated transfer function was not able to completely eliminate the noise present in the data.

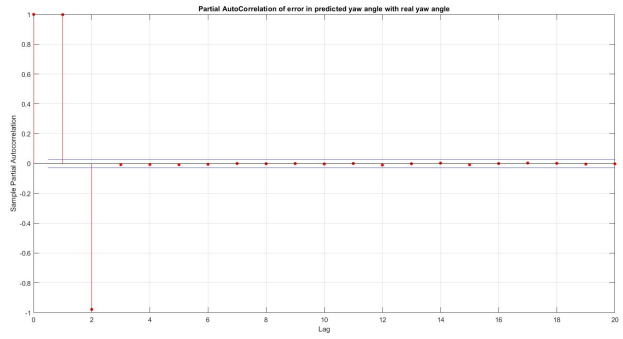



Figure 5.11: Error Partial Auto-Correlation Plot for ARX Model Estimation

On further finding the partial autocorrelation of the residual data we found that the current error was correlated up to (t-2) error terms behind.

Based on the significant correlation values between the error on predicted data and both itself and the input, it was determined that the residual correlation was due to noise in the system. Since the ARX model used in the initial system identification process could not account for this noise, it was necessary to explore alternative modeling methods. As a result, the ARMAX model was tested as an alternative, which is capable of handling input and output noise in the system.

## 5.4 ARMAX Model

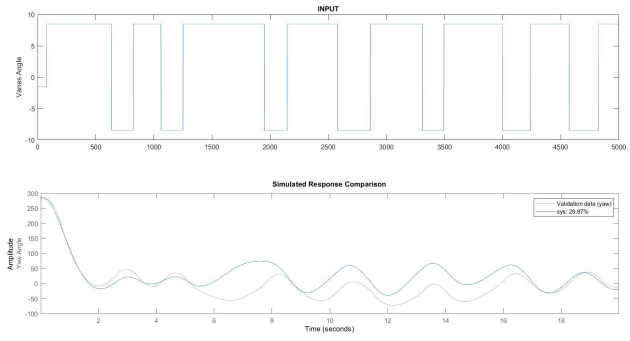



Figure 5.12: Input and Output Comparison Plot for ARMAX Model Estimation

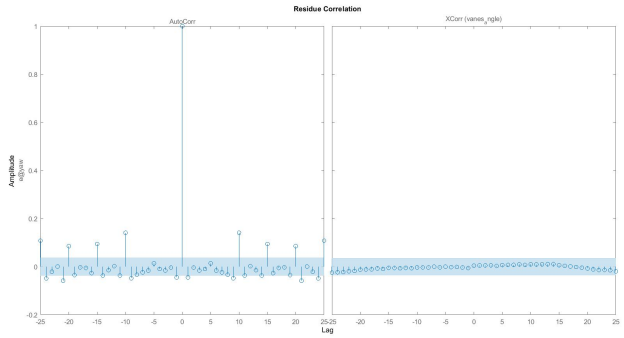



Figure 5.13: Error Correlation Plot for ARMAX Model Estimation

Although the use of the ARMAX method eliminated the issue of significant residual correlation in the error terms, it resulted in a substantially decreased fitness of the estimated model compared to the ARX method. One possible explanation for this observation is that while the error data were highly correlated in the ARX model, their magnitude was relatively small, and therefore, had little impact on the model's fitness. However, by introducing additional parameters to the ARMAX model, the model may have become overparameterized, leading to a decrease in the model's performance. Additionally, the added complexity of the ARMAX model can lead to a decrease in the accuracy of the model fit, especially if the available data is limited.

## 5.5 Subspace Method

The result in both of the below cases are quite similar. A 3 order state space model was predicted for both cases which very well fitted the output data on validation data. Similarly the residuals of the predicted and actual noise had insignificant auto correlation and cross correlation with the given inputs.

### 5.5.1 Keeping ESC value in equilibrium

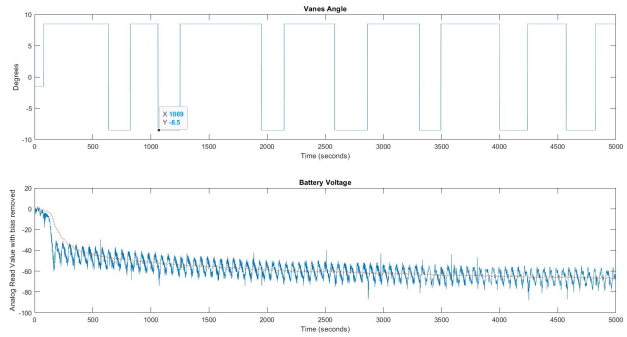



Figure 5.14: Input Plot for State Space Model Estimation with motor at equilibrium

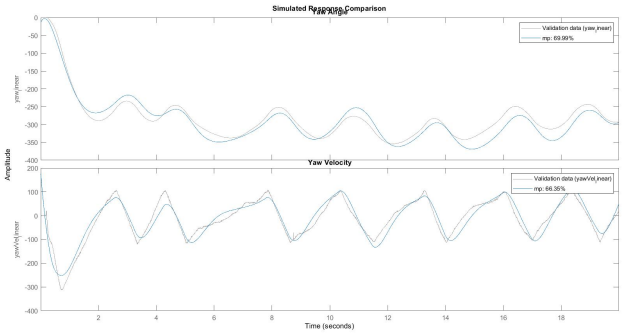



Figure 5.15: Output Comparison Plot for State Space Model Estimation with motor at equilibrium

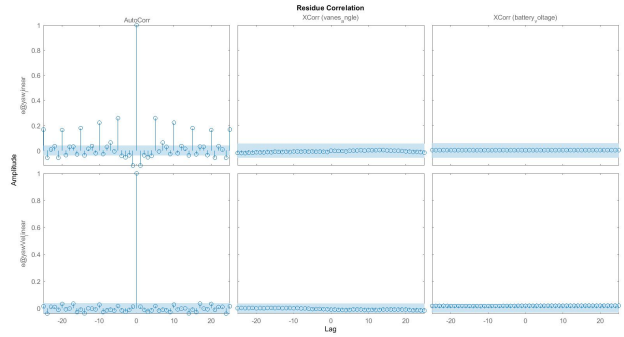



Figure 5.16: Error Correlation Plot for State Space Model Estimation with motor at equilibrium

### 5.5.2 Keeping vanes angle in equilibrium

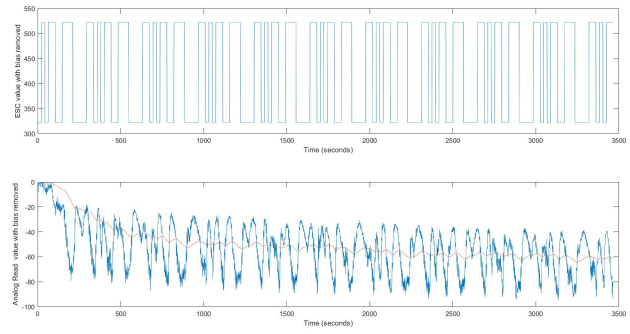



Figure 5.17: Input Plot for State Space Model Estimation with vanes at equilibrium

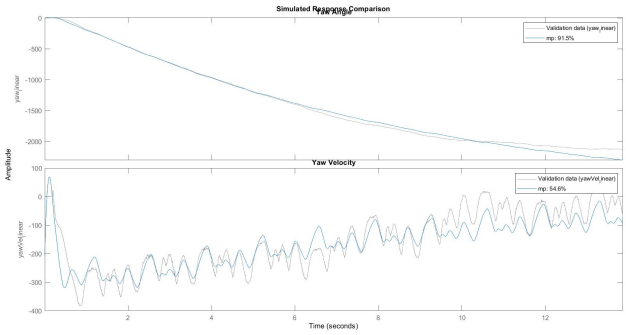



Figure 5.18: Output Comparison Plot for State Space Model Estimation with vanes at equilibrium

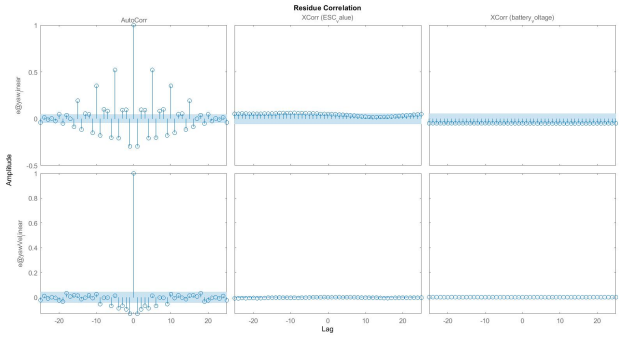



Figure 5.19: Error Correlation Plot for State Space Model Estimation with vanes at equilibrium

## 5.6 Comparison Between ARX, ARMAX and Subspace Method

response did not align with the physical model's behavior. Specifically, the model's prediction indicated that the yaw angle would settle to a constant value in response to a step input, whereas in reality the physical model would continue to increase its yaw angle in one direction. This discrepancy in the model's prediction can be attributed to the short duration of time over which the system was excited during experimentation. The limited battery capacity prevented us from extending the experimentation time, resulting in insufficient data to properly verify the model's accuracy. In order to address this issue, longer experimentation times could be utilized in future studies to ensure that the data captures the full range of the system's dynamics. This approach could provide more accurate modeling results and improve the overall effectiveness of the control system design.

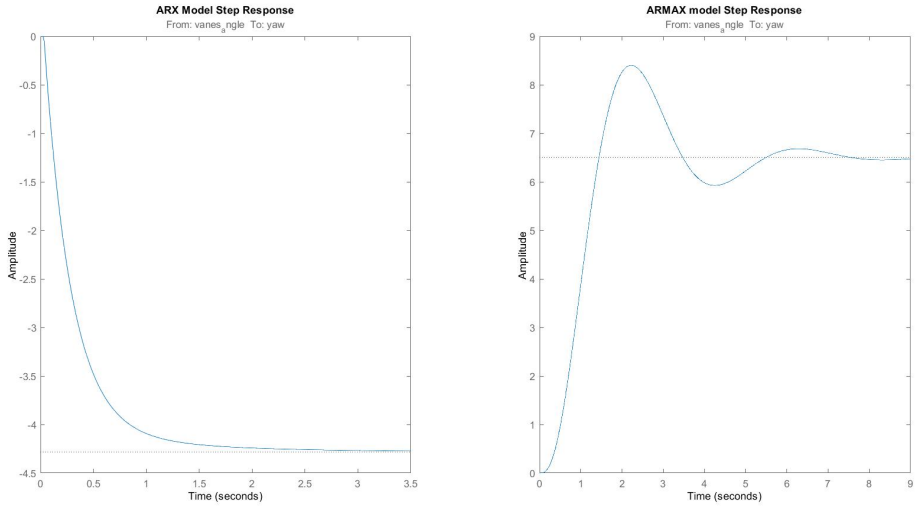



Figure 5.20: Step Response of ARX and ARMAX Model

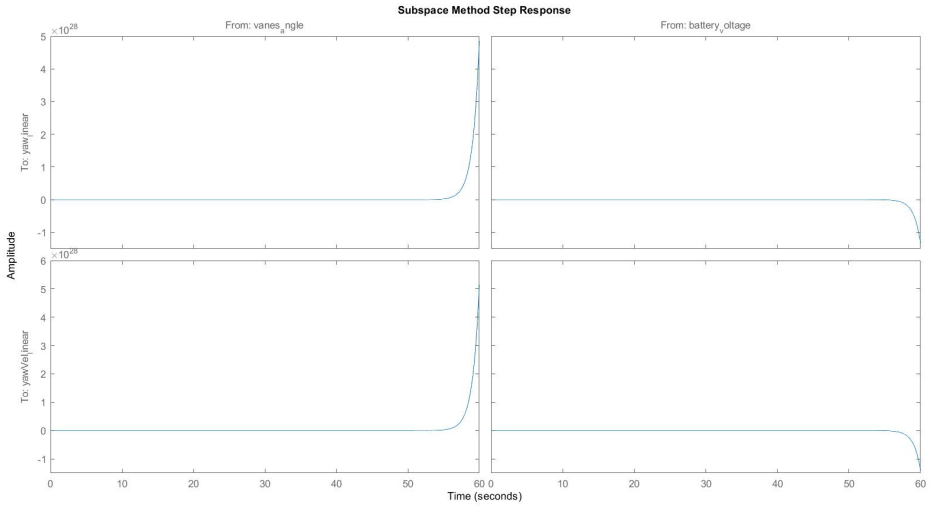



Figure 5.21: Step Response of State Space Model

Compared to the ARX and ARMAX models, the subspace method provided a more accurate approximation of the model's behavior. Specifically, the step response of the subspace model indicated that the yaw angle and yaw velocity would continue to increase in response to a step input, which aligned with the physical model's actual behavior. One possible explanation for the superior performance of the subspace method is that it takes into account the effect of the battery voltage, which was used as an input during experimentation. This additional information likely improved the model's accuracy by providing a more complete representation of the system's behavior.

Overall, the subspace method demonstrated significant advantages over other modeling

approaches in this case, highlighting the importance of carefully considering model selection and input/output data when designing control systems.

## 5.7 Open Loop Altitude Control

The simulation was done in matlab simulink

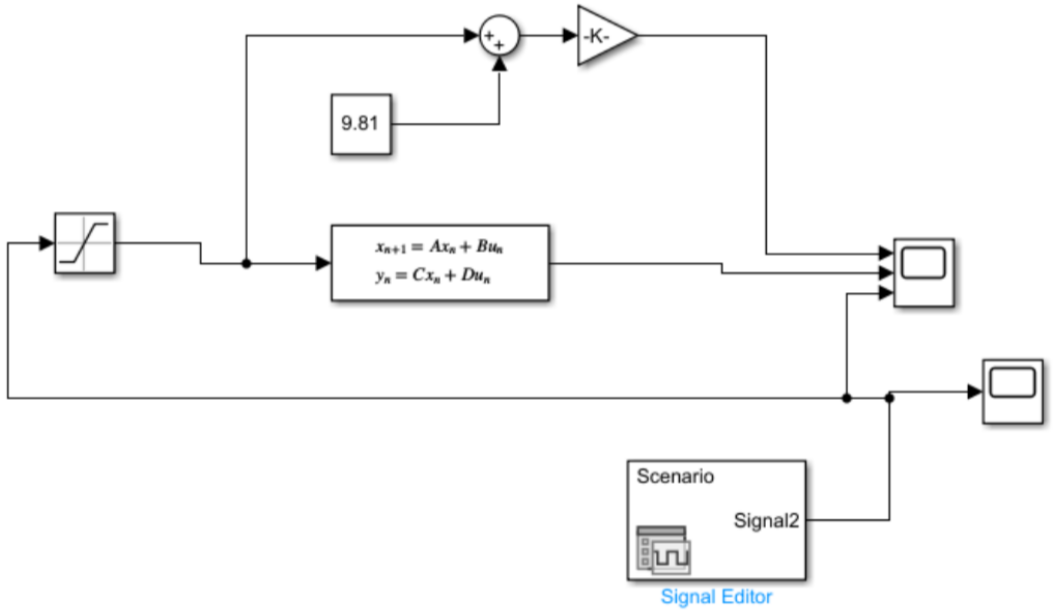



Figure 5.22: Altitude Control Block Diagram

In the simulation, a discrete-time state space block was used which represented the transfer function from first principal corresponding to altitude control of the rocket. A signal editor block was used to generate input signals. The input was passed through the saturator which set the boundaries to the input and the output from the saturator block was given to the model. The input to the model and output from the model were plotted using a scope.

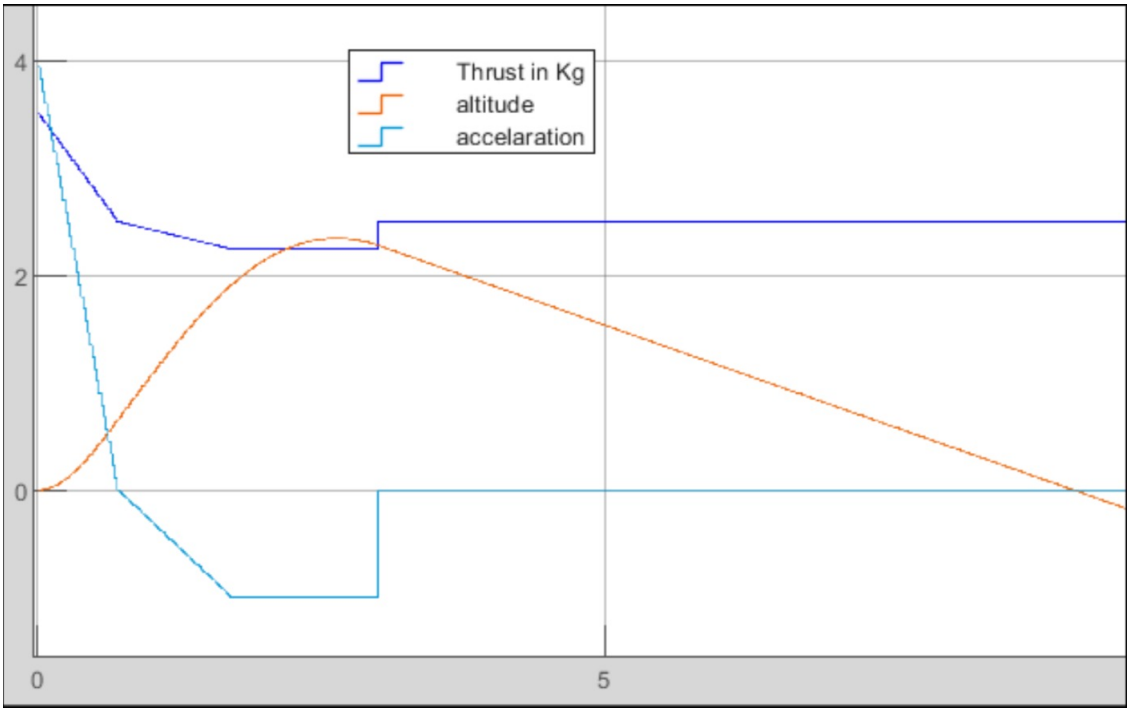



Figure 5.23: Simulation Result of Altitude Control

The above figure shows simulation results of open loop altitude control. Apogee was reached in approximately 2.5 seconds. Landing took approximately 7 seconds in addition. Total flight time was about 9 seconds.

## 5.8 Simulation of the LQG controller With Thrust Vectoring Nozzle

To find the optimal feedback matrix ($K$) a simulation environment was set up in simulink using the state space model obtained from first principle. The values of $Q$ and $R$ were tuned to adjust $K$ until the required response from the simulation was obtained. Integral action and set-point tracking were also analyzed.

The noise was also added in the simulation after properly measuring all the sensors and measurement noise in the vehicle. Noise was present in the battery voltage measurement which resulted in noise in thrust measurement as thrust measurement relied on battery voltage. The signal to noise ratio of the BMP280 sensor was very high which was somehow reduced by using its internal moving average filter. Noise present in MPU6050 was also measured.

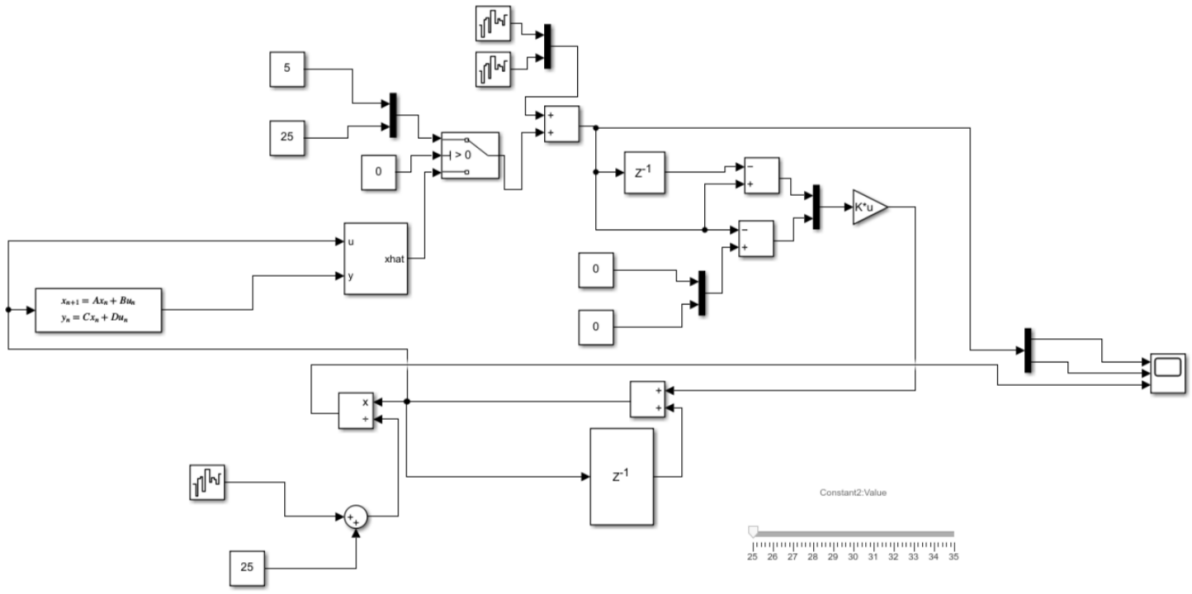



Figure 5.24: Attitude Control Simulation Block Diagram

The vehicle's initial states were set to 5 degrees tilt with zero angular velocity and the simulation was run. There was some noise in the states of the vehicle in spite of using a kalman filter which was due to the noise in the input thrust. The nozzle angle was very less which solved the burden of using constraints in the controller design. Small overshoot in both states and input was also seen in the simulation. The vehicle took around 2 seconds to come to a steady state which met our requirements.

## 5.9 Result using LQG and Thrust Vectoring Nozzle

The rocket took off satisfactorily and reached an apogee of about 1.5 meters taking approximately 2.5 seconds of time. Landing could not be achieved. Lack of yaw control mechanism is the probable reason for it. The figure below shows change in altitude with time.

The vehicle was fairly straight initially as shown in the graph below. After the first second, due to lack of yaw control wobbling and fast rotation in the yaw axis, our small angle approximation failed thus making the vehicle unstable.

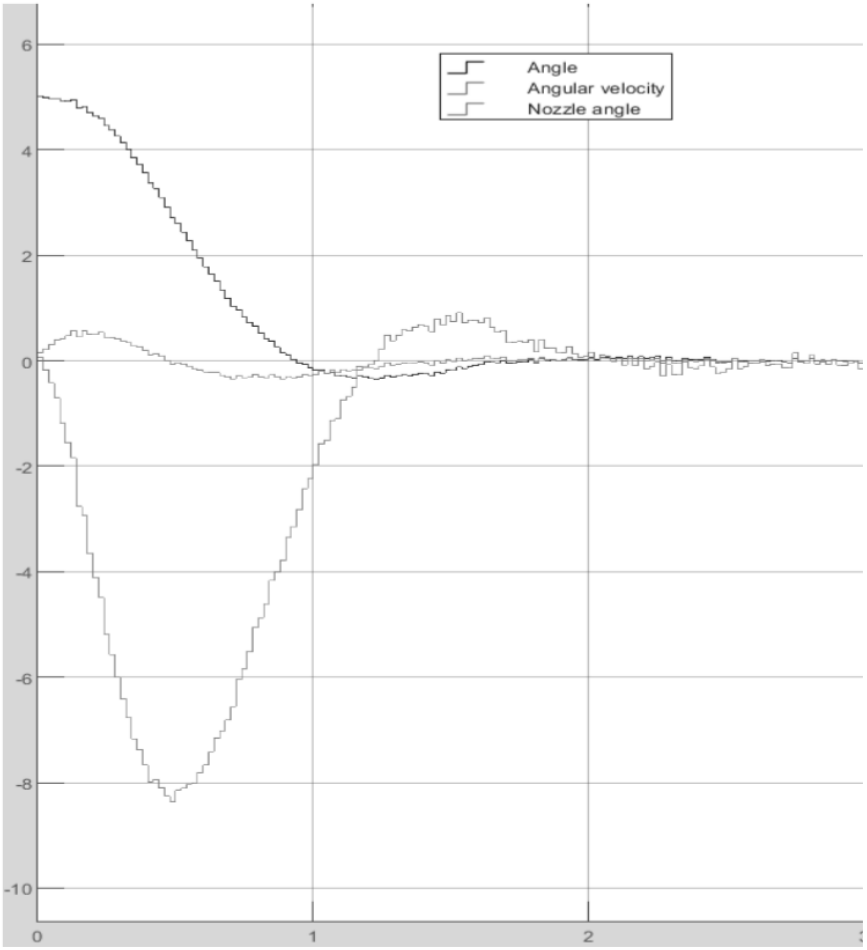



Figure 5.25: Attitude Control Simulation Result

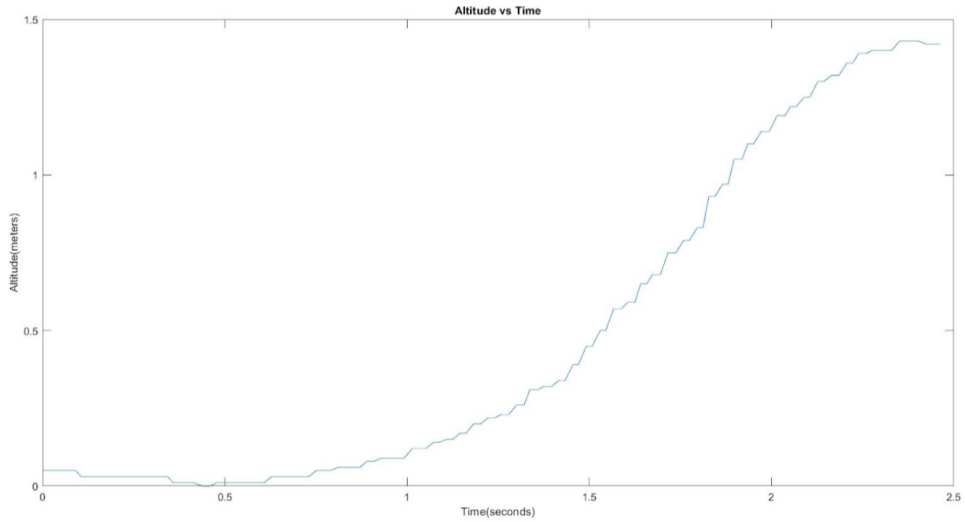



Figure 5.26: Altitude Vs Time

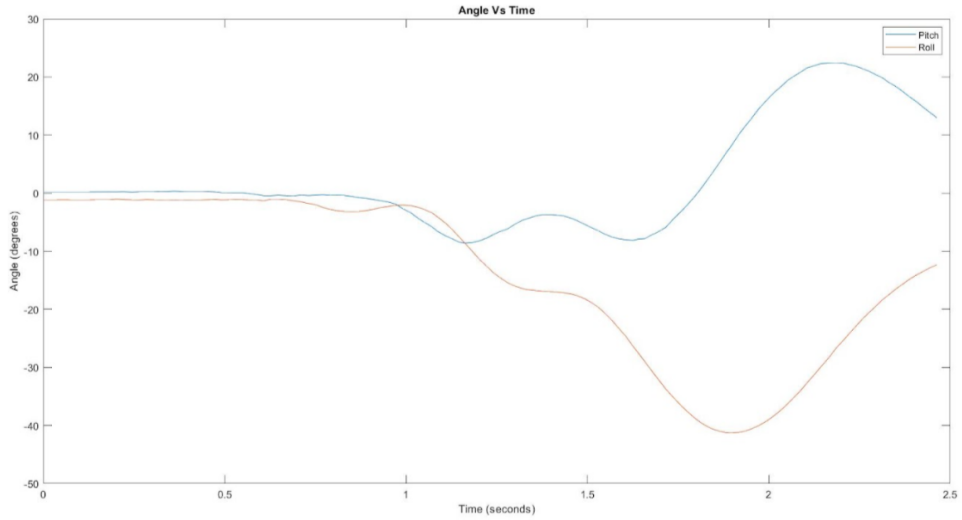



Figure 5.27: Angle Vs Thrust

The angular velocity changed as shown below.

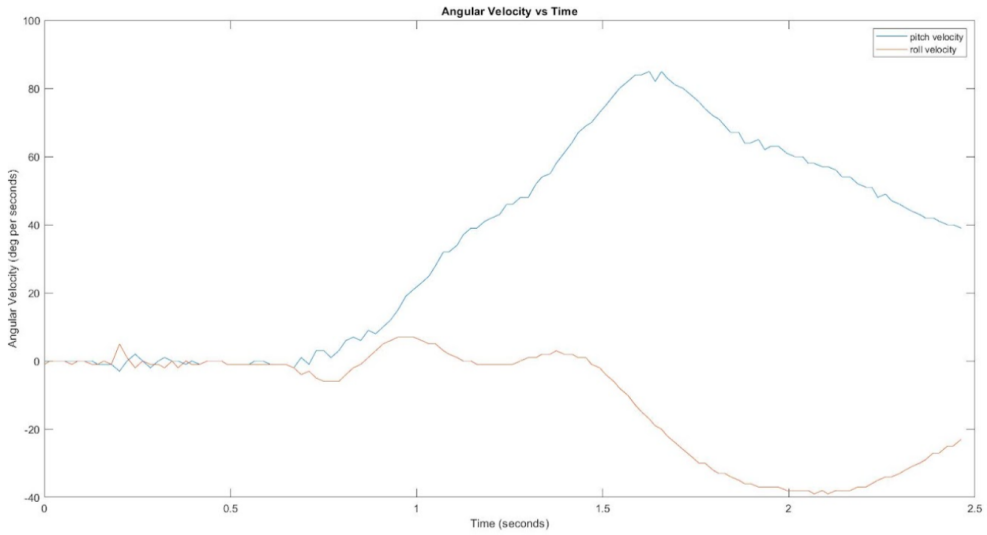



Figure 5.28: Angular velocity Vs Time

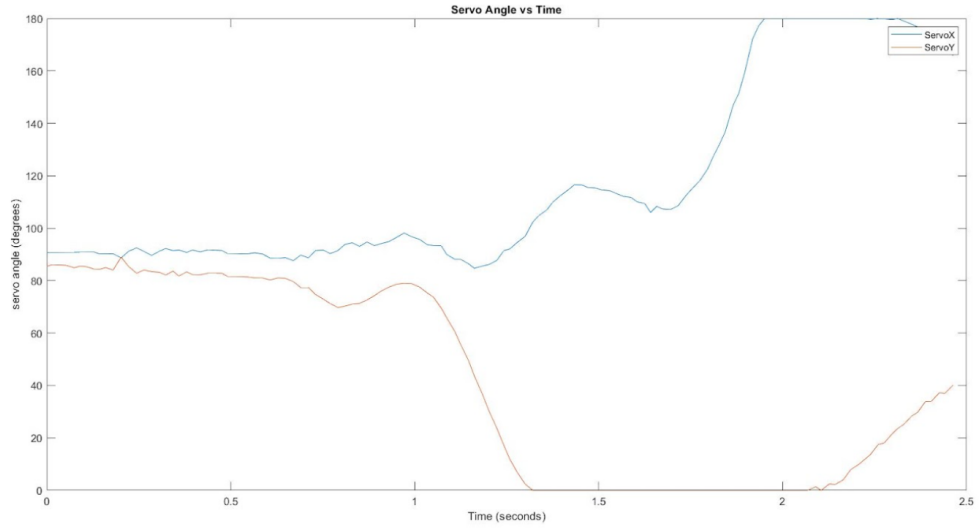



Figure 5.29: Servo Angle Vs Time

The above servo angle vs time also points out the fact that the vehicle was fairly stable in the first second and became unstable after that.
The change in throttle with time is shown below:

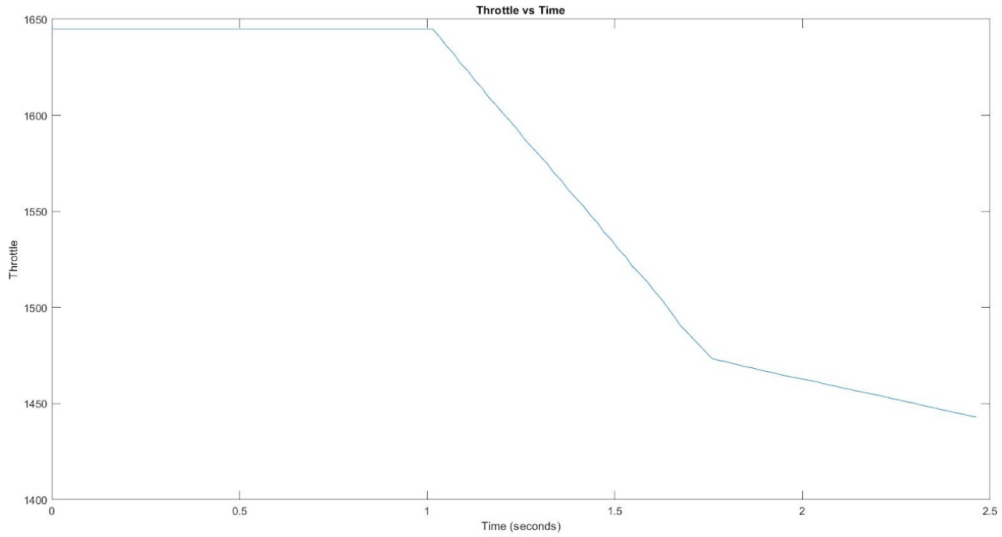



Figure 5.30: Throttle Vs Time

## 5.10 Simulation of the PID controller with Vanes

To simulate the PID controller in MATLAB, the data obtained from the subspace method was converted to a transfer function, where the input was the vane angle and the output was the yaw angle. The resulting transfer function was then used with a discrete PID controller to obtain the final results.

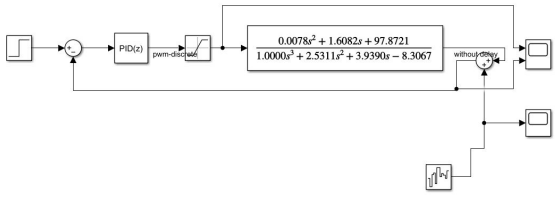



Figure 5.31: PID Simulation Block Diagram with Vanes

## 5.11 Result of the PID controller with Vanes

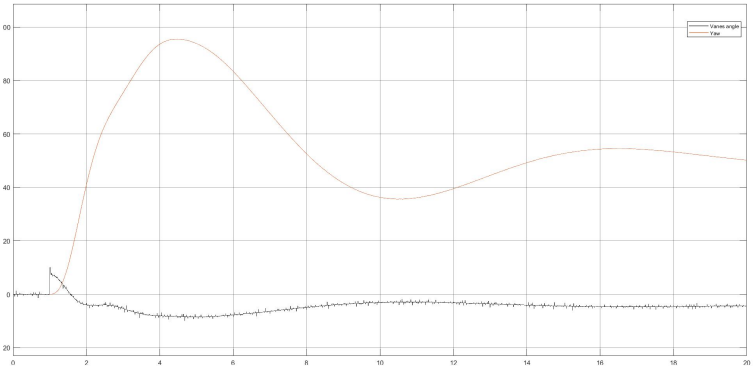



Figure 5.32: PID Simulink Graph ($AutoTune$)($Vanesangle \pm 10°$)

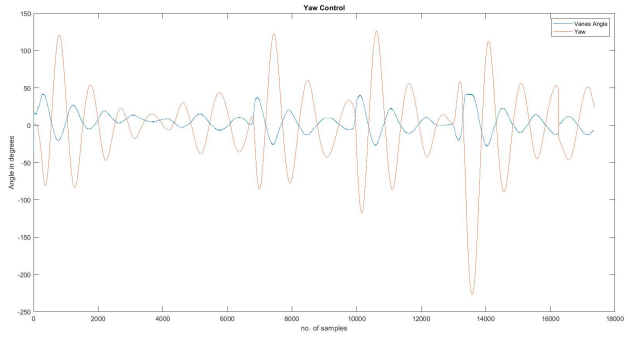



Figure 5.33: PID Simulation Graph ($Vanesangle \pm 40°$)

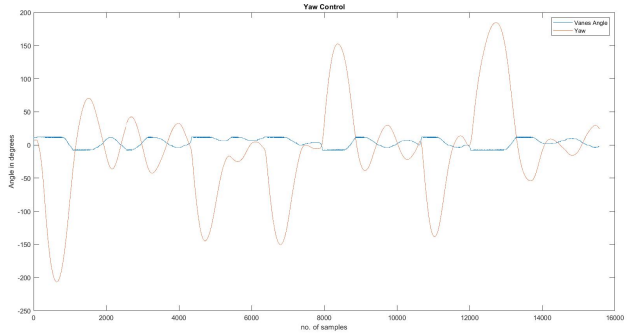
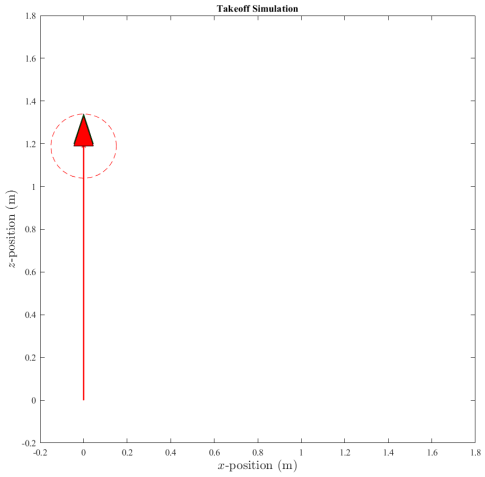



Figure 5.34: PID Simulation Graph ($Vanesangle \pm 10°$)

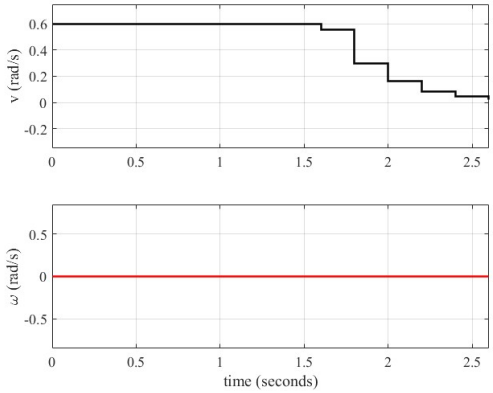
The simulation results showed significant overshoot and required a considerable amount of time to reach a steady state. The simulation was conducted with the vanes angle limit set to a maximum of 10 degrees in both directions. The experimental results were consistent with

the simulation, as the rocket was successfully controlled in the yaw direction. /par However, upon changing the vanes angle limit to 40 degrees, the same PID control parameter caused the rocket to become very reactive to disturbances. While the output remained stable, it became oscillatory with increased overshoot and settling time.

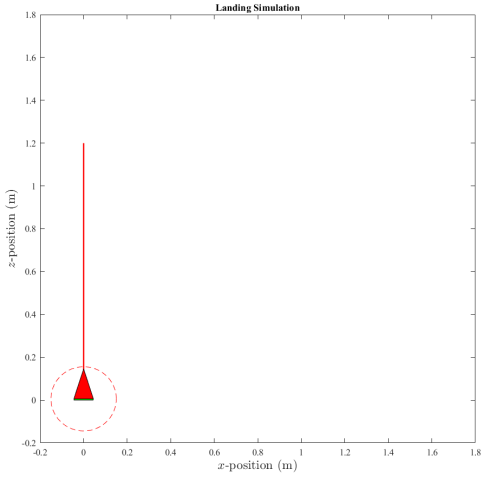
## 5.12 Simulation Result of the MPC controller



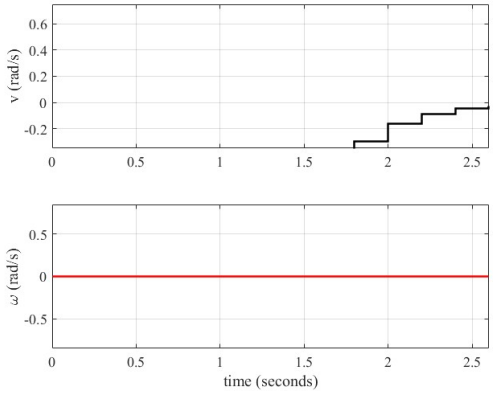

(a) MPC Single Shooting Takeoff Animation



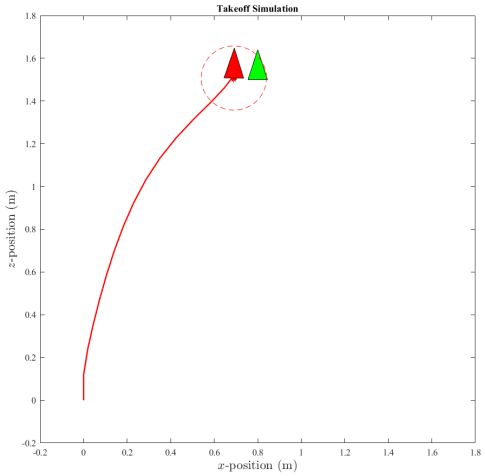

(b) MPC Single Shooting Takeoff Graph



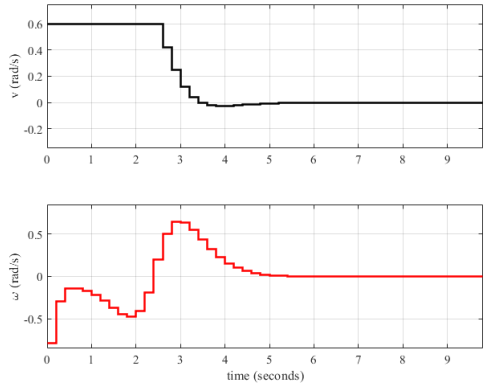

(a) MPC Single Shooting Landing Animation



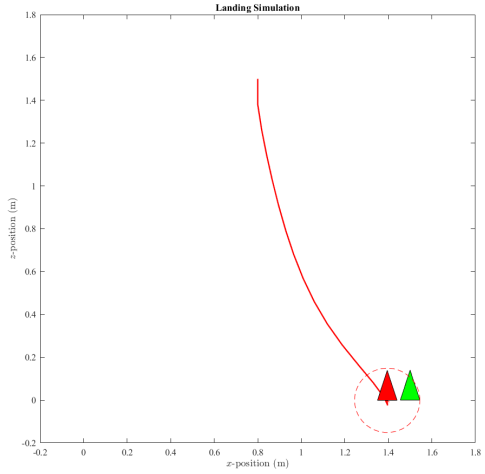

(b) MPC Single Shooting Landing Graph



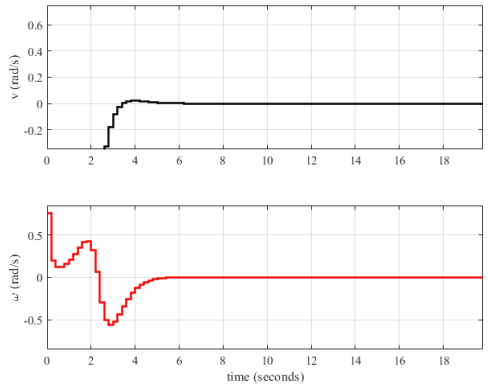

(a) MPC Single Shooting Takeoff Animation



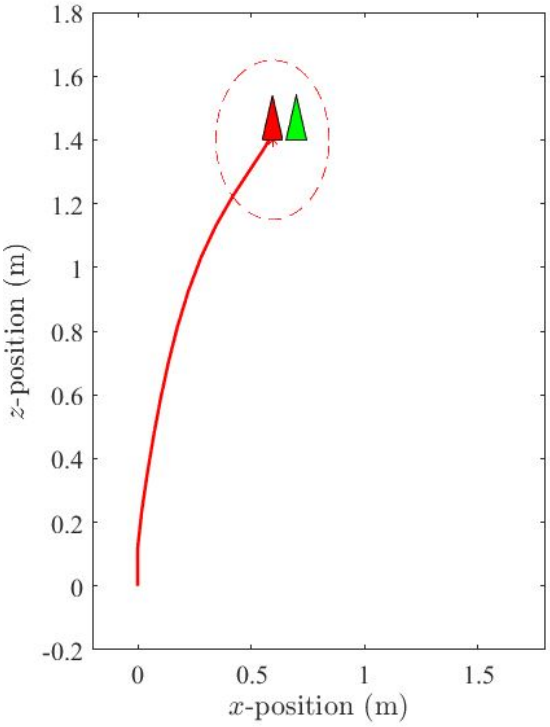

(b) MPC Single Shooting Takeoff Graph



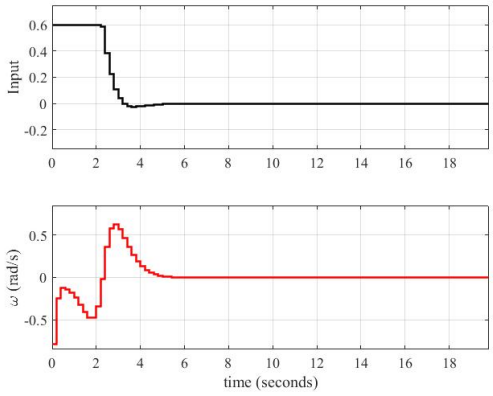

(a) MPC Single Shooting Landing Animation



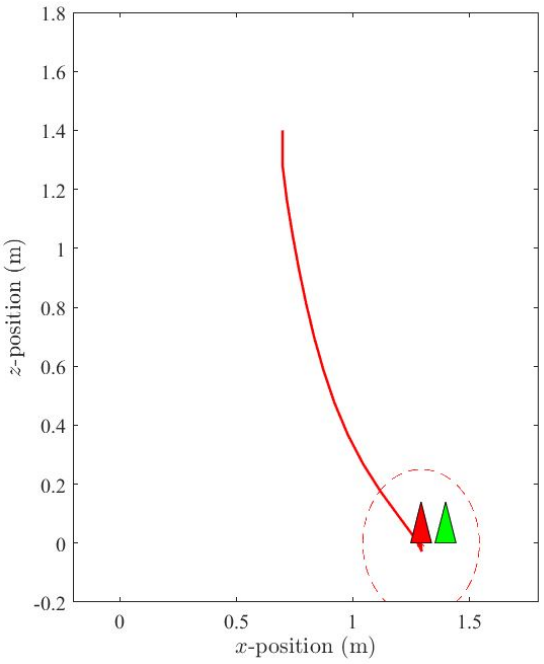

(b) MPC Single Shooting Landing Graph



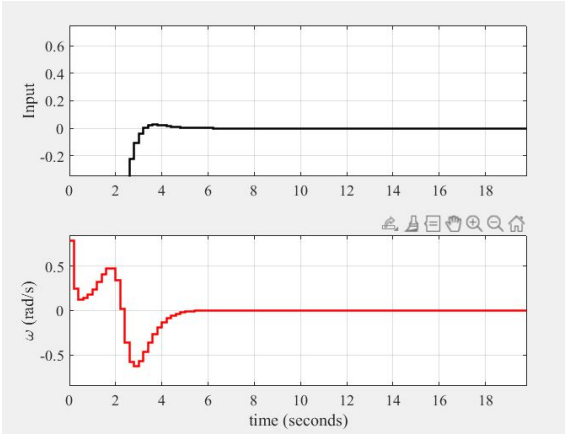

(a) MPC Multiple Shooting Takeoff Animation




(b) MPC Multiple Shooting Takeoff Graph




(a) MPC Multiple Shooting Landing Animation




(b) MPC Multiple Shooting Landing Graph

The MPC controller algorithm was tested using a very simplistic model. The parameters Q matrix and R matrix were tweaked along with the prediction horizon. Both single shooting and multiple shooting approach were tested, The multiple shooting approach turned out to be 20 times faster than the single shooting approach. Now the task remaining is to change the state space matrix in the MPC algorithm with the one determined from the subspace method with a proper state observer and simulate the response.

# 6. Conclusions

Based on the progress made so far, it can be concluded that the project is well underway towards achieving its objectives. The completion of sensor fusion for navigation and the setup of the simulation environment are significant milestones that have been achieved. The successful implementation of these steps will serve as the foundation for developing and testing the control system for a VTOL vehicle using Model Predictive Control (MPC) and Linear Quadratic (LQ) algorithms.

With these initial stages complete, the project team can move on to the next phase, which involves implementing the MPC algorithms in the simulated environment and optimizing the control system parameters to achieve precise guidance, navigation, and control over a predetermined trajectory. Once this phase is completed, the team can then proceed to test the LQG control system's performance in a real-world environment and evaluate its effectiveness in enhancing the VTOL vehicle's flight characteristics.

Thus, the progress made so far is encouraging, and the team is well on its way towards achieving the project's objectives. The successful completion of this project will not only demonstrate the feasibility and effectiveness of using LQ and MPC algorithms in the design and control of a VTOL vehicle, but it will also contribute to the ongoing efforts to enhance the capabilities of space vehicles and advance the field of aerospace engineering.

# 7. Limitations and Future enhancement

## 7.1 Limitations

- The current project only focuses on the use of Model Predictive Control and Linear Quadratic algorithms for the design and control of a VTOL vehicle. Other control algorithms and techniques were not explored, which could limit the overall performance and effectiveness of the control system.

- The testing of the MPC control system's performance in a real-world environment was limited due to safety concerns and logistical constraints. Thus, the evaluation of the control system's effectiveness in enhancing the VTOL vehicle's flight characteristics may not be fully representative of its actual capabilities.

## 7.2 Future Enhancements

- Utilizing Reinforcement Learning (RL) algorithms for the control of the rocket to enhance its overall performance and efficiency.

- Investigating the use of other advanced control algorithms and techniques, such as Nonlinear Model Predictive Control (NMPC) and Adaptive Control, to further optimize the design and control of the VTOL vehicle.

- Conducting more extensive testing and evaluation of the control system's performance in real-world environments, utilizing more sophisticated sensors and instrumentation to provide a more accurate assessment of its capabilities.

# References


[1] Shekhar Pathak, Sameer Agarwal, Sohith Kakumanu, Siddharth Sudhakar, and G Srinivas. Design and implementation of flight computer for sounding rockets-s3fc. In *2022 6th International Conference on Electronics, Communication and Aerospace Technology*, pages 164–170. IEEE, 2022.

[2] Lars Yndal Sørensen, Lars Toft Jacobsen, and John Paulin Hansen. Low cost and flexible uav deployment of sensors. *Sensors*, 17(1):154, 2017.

[3] Jurek Z Sasiadek. Sensor fusion. *Annual Reviews in Control*, 26(2):203–228, 2002.

[4] JZ Sasiadek, Q Wang, and MB Zeremba. Fuzzy adaptive kalman filtering for ins/gps data fusion. In *Proceedings of the 2000 IEEE International Symposium on Intelligent Control. Held jointly with the 8th IEEE Mediterranean Conference on Control and Automation (Cat. No. 00CH37147)*, pages 181–186. IEEE, 2000.

[5] Ujjval N Patel and Imraan A Faruque. Sensor fusion to improve state estimate accuracy using multiple inertial measurement units. In *2021 IEEE International Symposium on Inertial Sensors and Systems (INERTIAL)*, pages 1–4. IEEE, 2021.

[6] Uğur Kayasal. Modeling and simulation of a navigation system with an imu and a magnetometer. Master's thesis, Middle East Technical University, 2007.

[7] Qigao Fan, Hai Zhang, Peng Pan, Xiangpeng Zhuang, Jie Jia, Pengsong Zhang, Zhengqing Zhao, Gaowen Zhu, and Yuanyuan Tang. Improved pedestrian dead reckoning based on a robust adaptive kalman filter for indoor inertial location system. *Sensors*, 19(2):294, 2019.

[8] Manan S Gandhi, Lee Whitcher, Evangelos Theodorou, and Eric N Johnson. Practical system identification for small vtol unmanned aerial vehicle. In *AIAA Scitech 2019 forum*, page 1982, 2019.

[9] Pietro Panizza, Fabio Riccardi, and Marco Lovera. Black-box and grey-box identification of the attitude dynamics for a variable-pitch quadrotor. *IFAC-PapersOnLine*, 48(9):61–66, 2015.

[10] Christopher E Hann, Malcolm Snowdon, Avinash Rao, Robert Tang, Agnetha Korevaar, Greg Skinner, Alex Keall, XiaoQi Chen, and J Geoffrey Chase. Rocket roll dynamics and disturbance—minimal modelling and system identification. In *2010 11th International Conference on Control Automation Robotics & Vision*, pages 1736–1741. IEEE, 2010.

[11] Michael A Johnson and Mohammad H Moradi. *PID control*. Springer, 2005.

[12] A O'Dwyer. Pi and pid controller tuning rules for time delay processes: a summary. *Technical report AOD-00-01*, 2000.

[13] Aidan O'Dwyer. A summary of pi and pid controller tuning rules for processes with time delay. part 1: Pi controller tuning rules. *IFAC Proceedings Volumes*, 33(4):159–164, 2000.

[14] Karl Worthmann, Mohamed W Mehrez, Mario Zanon, George KI Mann, Raymond G Gosine, and Moritz Diehl. Model predictive control of nonholonomic mobile robots without stabilizing constraints and costs. *IEEE transactions on control systems technology*, 24(4):1394–1406, 2015.

[15] Carlo Alberto Pascucci, Samir Bennani, and Alberto Bemporad. Model predictive control for powered descent guidance and control. In *2015 European Control Conference (ECC)*, pages 1388–1393. IEEE, 2015.

# Appendices