People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

University M'Hamed BOUGARA – Boumerdès

Institute of Electrical and Electronic Engineering

Department of Electronics

Project Report Presented in Partial Fulfilment of

the Requirements of the Degree of

'MASTER'

In Control Engineering

Modelling, Controller Design and Implementation of a Small-scale 3DOF Helicopter Using LabVIEW and STM32F4xx Microcontroller.

Presented By:

- **DEBILOU Abdelalim**
- **MANAMANI Zineddine**

Supervisor:

DR.Abderrahmane OUADI

Registration Number:……../2023

Abstract

The 3DOF is a non-linear underactuated system. It is inspired from the Quanser 3DOF helicopter, however, the materials used and the dimensions as well as the motors are different therefore a model has to be defined and improved from previous works. The 3DOF small-scale helicopter is a map of a real CHINOOK helicopter. The aim of this work is to create a platform for testing and deploying the different controllers as it will serve as an educational and research platform. Both linear and nonlinear system identification techniques were used to derive a model. The nonlinear model was linearized around the region of operation and then an LQR controller was designed. The redundancies such as the ADC for generating the control action were removed to reduce delay times and hardware costs. The previous students utilized the NI DAQ 6221 M series for system identification and during the running of the controller. In our case, the NI DAQ was used solely for identification as it facilitates the implementation, verification, visualization, and logging. while the STM32F4 microcontroller was used for data acquisition during control operation. Introducing the STM32 removed the shortage of hardware counters. An LQR controller was designed, tuned, and tested. Finally, a GUI for logging the data in order to evaluate the performance of the controlled process was designed.

Key terms: 3DOF, Reasearch platform, Modelling, feedback linearization, LQI, STM32F4, LabVIEW, Quadrature encoders, Jacobian linearization.

Dedication

This humble work is dedicated to our parents whose unwavering support and encouragement have been the cornerstone of our academic journey. Also, to our brothers and sisters and all the family members.Last but not least, to our friends whose friendship and camaraderie have made the journey all the more enjoyable.

Acknowledgments

The successful completion of this master project would not have been possible without Allah and his grace.
We would also like to express our sincere appreciation to our supervisor, DR.Abderrahmane OUADI for his invaluable guidance, expertise, and encouragement throughout the completion of this thesis. His unwavering commitment to excellence and his willingness to share his knowledge has been instrumental in shaping the outcome of this work. We are grateful for his patience, understanding, and constructive feedback, which have undoubtedly enhanced the quality of this project.

List of Abbreviations and Terms

IEEE	Institute of Electrical and Electronics Engineering
PWM	Pulse-Width Modulation
MSE	Mean square error
NLLS	Non linear least squares
MSE	Mean square error
FBD	Full body diagram
LQR	Linear quadratic regulator
PID	Proportional integral derivative controller
fminunc	function minimization unconstrained
ARR	Auto-Reload Register
PSC	Prescaler register
LabView	Laboratory Virtual Instrumentation Engineering Workbench
VISA	Virtual Instrument System Architecture
LQI	Linear quadratic regulator with integral action
DOF	Degrees of freedom
RPM	Revolution per minute
GUI	Graphical user interface
UART	universal asynchronous receiver transmitter

Table of Contents

Abstract . . . . **i**

Dedication . . . . **ii**

Acknowledgments . . . . **iii**

List of Abbreviations and Terms . . . . **iv**

Table of Contents . . . . **vii**

List of Figures . . . . **viii**

List of Tables . . . . **x**

General Introduction **1**

1 Model Identification **4**

1.1 Tools used for system identification 4

1.1.1 NI PCI 6221 4

1.1.2 MATLAB 4

1.2 The helicopter system 5

1.3 Safety measures 6

1.4 Thrust identification 6

1.5 Physical measurments 7

1.6 Modeling the three axes 7

1.6.1 The pitch Axis 8

1.6.2 Elevation Axis 10

1.6.3 Travel Axis 14

1.7 State-space model derivation 18

1.8 Conclusion: 19

2 Controller design **20**

2.1 System linearization 20

2.1.1 Calculating the equilibrium point 20

2.1.2 Obtaining the linearized model 21

2.2 System characteristics 22

2.2.1 Decoupling the system 22

2.2.2 Stability analysis 23
2.2.3 Controllability analysis 24
2.2.4 Observability analysis 25
2.3 Nonlinearities compensation using Feedback linearization for the elevation system 27
2.3.1 Feedback linearization for systems in the companion form 28
2.3.2 Compensating the nonlinearities in the elevation system: 28
2.4 State feedback controllers 29
2.4.1 Pole placement 30
2.4.2 LQR controller 31
2.5 LQR tunning 35
2.5.1 Tuning the LQR for the elevation 36
2.5.2 Tuning the LQR for the Travel 37
2.5.3 Tuning the LQR for the Pitch 38
2.6 Conclusion 38

3 Implementation 40
3.1 LABView and the NI acquisition board 40
3.2 The STM32F4 microcontroller 40
3.2.1 Hardware counters 41
3.2.2 Acquiring the three angles 42
3.2.3 Generating the PWM signals 42
3.2.4 Reading the maneuvering actions 42
3.2.5 Serial communication for the data and control action 42
3.3 Thrust unit 44
3.4 The quadrature encoder 45
3.5 The state diagram for maneuvering using the joystick 45
3.6 Determination of the sampling time 48
3.7 Implementing the controllers in labview 48
3.8 The user interface 50
3.9 Conclusion 51

4 Evaluation and discussion 52
4.1 Usage of the UART for real-time control 52
4.2 Controller Performance 52
4.3 Improvements 56

General Conclusion 58

A Appendix 60

A.1 Model identification 60
A.1.1 Code used in pitch identification 60
A.1.2 Codes used in elevation identification 60
A.1.3 Codes used in travel identification 61
A.2 Control design 62
A.2.1 Simulink code for simulating the elevation nonlinear system . . . 62
A.2.2 Simulink code for feedback linearization block. 63
A.2.3 Simulink code for Comparing the linear and nonlinear elevation systems. 63
A.2.4 Matlab code for tuning the LQR for the elevation. 63
A.2.5 Matlab code for tuning the LQR for the travel. 64
A.3 Implementation: 64
A.3.1 STM code used for data acquisition, PWM generation and communication 64
A.3.2 Interrupt service routines for the acquisition and control synchronization 65
A.3.3 Simulink code for simulationg the LQR controller for the elevation system: 65
A.3.4 Simulink code for simulation of the LQR controller for the travel system: 66

References **67**

List of Figures

0.1 Boeing HC-1B Chinook . . . 1
0.2 Quanser 3-DOF Helicopter rig . . . 1
0.3 3-DOF test bench at IGEE, (1) computer, (2) Power supply, (3) Encoder, (4) DAQ-board, (5) Motors: front on the left, (6) Motor+Propeller, (7) ESC, (8) Counter-weight. . . . 2

1.1 A basic schematic of the helicopter system[5] . . . 6
1.2 Stand for identifying the thrust unit . . . 7
1.3 FBD of the pitch axis[5] . . . 8
1.4 Recorded pitch response . . . 9
1.5 Comparison between simulated and experimental pitch response . . . 10
1.6 FBD of the Elevation axis[5] . . . 11
1.7 Thrust required at each elevation . . . 12
1.8 Recorded elevation response and the inputs used . . . 13
1.9 Experimental elevation data vs simulated model response . . . 14
1.10 FBD of the travel axis [5] . . . 14
1.11 The recorded travel response and the input used . . . 16
1.12 Actual response versus the simulated one. . . . 17
1.13 Travel model validation . . . 17

2.1 The linear and non-linear elevation state space response to the same input. 27
2.2 The linear and non-linear elevation state space response to the same input after using feedback linearization. . . . 29
2.3 a basic schematic of the state feedback controller.[10] . . . 30
2.4 LQR with forward gain[10] . . . 33
2.5 LQR with integral action[10] . . . 34
2.6 Bode plot of the closed-loop elevation system. . . . 36
2.7 Bode plot closed-loop travel system. . . . 36

3.1 Simplified diagram showing the operation of a general purpose timer . . . 41
3.2 General architecture of the hardware configuration. . . . 43
3.3 The thrust unit devices . . . 44
3.4 Sensor used in the 3DOF for the angle measurement . . . 45
3.5 FlowChart for activating the pitch and the travel controllers when using the joystick . . . 46

3.6 LabVIEW code for activating pitch and the travel controllers when using the joystick 47
3.7 LabVIEW diagram for the elevation controller 49
3.8 LabVIEW diagram for the travel controller 49
3.9 LabVIEW diagram for the pitch controller 50
3.10 Graphical user interface for the control phase 51

4.1 Elevation axis responses 53
4.2 Travel axis response 53
4.3 Response to a setpoint change in the pitch axis 54
4.4 Elevation Disturbance Response 55
4.5 Travel disturbance response 56

A.1 System identification code 60
A.2 The Matlab code for the NLLS algorithm 60
A.3 The code of the included simulated response function. 61
A.4 The code of simulating the nonlinear elevation equation. 61
A.5 The code of used for travel identification 61
A.6 The code used to estimate the travel parameters. 62
A.7 Simulink code for simulating the elevation nonlinear system 62
A.8 Simulink code for feedback linearization block. 63
A.9 Simulink code for Comparing the linear and nonlinear elevation systems. . 63
A.10 Matlab code for the tuning of LQR for the elevation system. 63
A.11 Matlab code for the tuning of LQR for the travel system. 64
A.12 STM code used for data acquisition, PWM generation and communication 64
A.13 Interrupt service routines for the aqcuisition and control synchronization . 65
A.14 Simulink code for simulation the LQR controller for the elevation system: 65
A.15 Simulink code for simulating the LQR controller for the travel system: . . 66

List of Tables

1 Physically measured quantities 7
2 Pitch equation identified parameters 10
3 Elevation equation identified parameters 13
4 Travel identified parameters 16

5 STM32F4 timers setup 41

General introduction

Background

Throughout history, the principles of flight and the designing of aircraft and helicopters have been extremely interesting topics to mankind. And the huge technological development of computers and control theory is one of the important factors in the development of new flying vehicles.
In this project, the 3-DOF helicopter rig will be used to examine the dynamics and control of a tandem rotor helicopter shown in figure 0.1 .In spite of its simplicity, this 3DOF-rig provides a good simulation of a tandem helicopter in real life for testing the effectiveness of various control theories.

Figure 0.1. Boeing HC-1B Chinook

Many have already built such a helicopter rig, especially The Quanser 3DOF helicopter[1]shown in figure 0.2, which is being used in control engineering courses at universities all over the world.

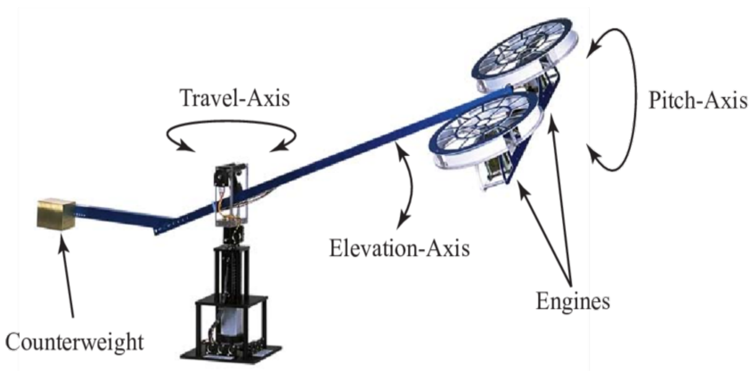


Figure 0.2. Quanser 3-DOF Helicopter rig

System description

Figure 0.3 illustrates the 3-DOF Helicopter system, which consists of a rectangular frame with two motors mounted at either end that power two propellers. The motors are aligned parallel to each other and generate thrust perpendicular to the frame. The helicopter frame is suspended from a joint on a long arm that can pitch around its center. The arm is attached to another instrumented joint that allows the helicopter body to move in the elevation and travel axes. A counterweight is placed at the other end of the arm to ensure that the effective mass of the helicopter is light enough to be lifted by the motors. This system is similar to a tandem rotor helicopter that is oriented perpendicular to the support arm.

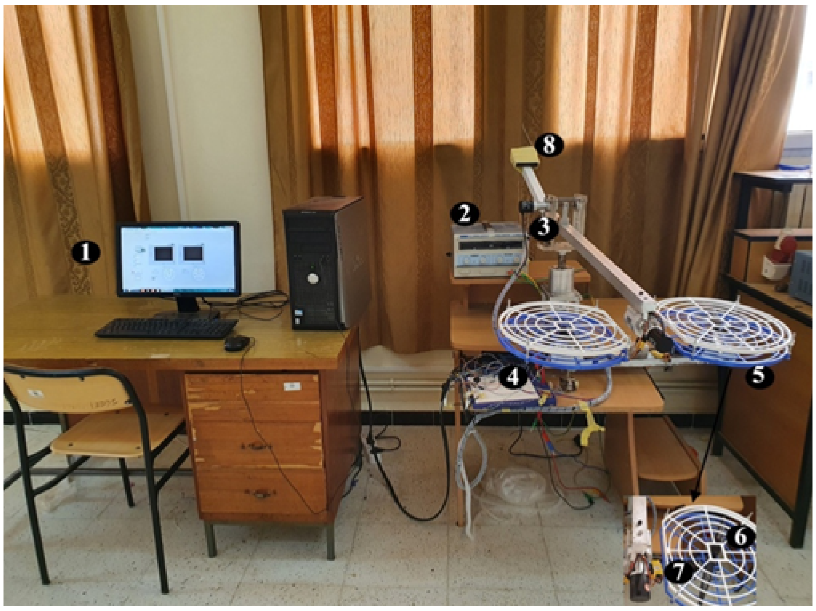


Figure 0.3. 3-DOF test bench at IGEE, (1) computer, (2) Power supply, (3) Encoder, (4) DAQ-board, (5) Motors: front on the left, (6) Motor+Propeller, (7) ESC, (8) Counterweight.

Applying a positive voltage to the front motor causes a positive pitch, while applying a positive voltage to the back motor results in a negative pitch. Elevating the body is also possible with a positive voltage to either motor. If the body pitches, it causes the thrust vectors to move the body, leading to yaw of the arm. An eight-contact slip ring is fitted to the vertical base, which channels electrical signals to and from the arm and helicopter, eliminates tangled wires, reduces friction, and enables unlimited and unhindered travel.

Objective

The scope of our project is the development of a new aqcuisition technique using the STM32F4 microcontroller as well as the development of a mathematical model and a control system for this 3-DOF helicopter test bench. The performance of the control system should be good enough to safely control the helicopter in flight. The elevation system will be controlled to operate at the full range of angles while the travel system will be controlled to operate at the 0 elevation angle. A simulation model will also be developed to test the behavior of the model and control system before applying it. Finally, the hardware used and the performance of the controller will be assessed.

Preview

This thesis comprises four chapters, each exploring a distinct aspect of the topic. This is a quick preview of each of the chapters:

Chapter 01: Provides the annotations for the different physical parameters, and then the equations of motion are derived using Euler's law. After that, it will be possible to numerically identify the parameters of the grey model using data recorded during the execution of the identification experiments. Finally, a state space model for the system was derived to be used for designing the controller.

Chapter 02: The obtained state model from the previous chapter will be linearized around the region of operation, and then an LQR controller will be analyzed and tuned using Matlab Simulink.

Chapter 03: Presents the tools and methods that are used in order to implement the software around the STM32F4 with the test bench. An emphasis is placed on using the STM32 to achieve real-time acquisition of the process variables and full-duplex data transmission with LabVIEW VISA. The LQR controllers developed in Chapter 2 are implemented in the newly designed system.

Chapter 04: is the evaluation chapter which provides an assessment for the Effectiveness and the impact of the proposed hardware and controller in terms of their reliability and accuracy. A test of the controller was performed using the designed system along with the software implemented in LabVIEW. Finally, A set of experiments were executed in order to assess the performance of the controllers.

Chapter 1. Model Identification

In this chapter, the mathematical model representing the dominant system dynamics will be developed. This will give a qualitative intuition of the system and, primarily, the governing equations that will be selected for the controller design. The validity of the model is confirmed through a comparison between open-loop simulations and real-world experiments conducted on the actual plant.

1.1 Tools used for system identification

The system identification process was conducted using the NI PCI-6221[2] for data acquisition and Arduino uno[3] for generating the pulse-width modulation (PWM) signals. However, they will be both replaced by the STM32f4 microcontroller in the rest of the operations. This is because access to the STM32 microcontroller was not available at that time. The software used for identification were LabVIEW and MATLAB[4].

1.1.1 NI PCI 6221

The NI PCI 6221 is a multifunction data acquisition (DAQ) device developed by National Instruments. It is designed to provide high-performance data acquisition capabilities for various measurement and control applications. The PCI 6221 offers simultaneous analog input and output channels, along with digital input and output lines. With a sampling rate of up to 250 kS/s per channel and a resolution of 16 bits, it enables precise and accurate measurement of analog signals. The device also supports a wide range of input and output voltage ranges, making it suitable for handling various signal types. Additionally, the NI PCI 6221 is equipped with onboard memory for buffering acquired data, minimizing the risk of data loss. Its compatibility with the NI-DAQmx driver and LabVIEW software ensures seamless integration and enables users to easily configure and control the device. Overall, the NI PCI 6221 offers a reliable and versatile solution for data acquisition applications, empowering engineers and researchers to gather and analyze data with efficiency and accuracy.

1.1.2 MATLAB

MATLAB is a powerful software environment widely used in scientific, engineering, and mathematical fields. Developed by MathWorks, MATLAB provides a comprehensive set of tools for data analysis, visualization, and algorithm development. With its intuitive programming language and extensive library of functions, MATLAB enables users to solve complex mathematical problems, perform numerical computations, and develop advanced simulations. It offers a flexible and interactive environment that supports both numerical and symbolic computations, making it suitable for a wide range of applications. MATLAB's rich visualization capabilities allow users to create plots, charts, and 3D graphics to analyze and present their data effectively. Moreover, MATLAB's integration with other programming languages and hardware interfaces enhances its versatility and enables seamless collaboration with other tools and systems. Whether used in academia or industry, MATLAB continues to be a valuable resource for researchers, engineers, and scientists for its ability to streamline data analysis, modeling, and algorithm development processes.

1.2 The helicopter system

The system rotates around the travel and elevation axes. The rig is fixed in the mount at point O. From figure 1.1, there are three angles that describe the helicopter position, which are defined as:

- Pitch angle (ρ): The rotational angle of the helicopter with respect to the pitch joint's axis. When the helicopter is horizontal, it is defined as zero.

- Elevation angle (ϵ): The angle formed by the arm and the overall XY plane. when the arm is horizontal, it is defined as zero.

- Travel angle (τ): The rotational angle of the rig with respect to the global travel axis. At the sensor initialization angle, it is set to zero.

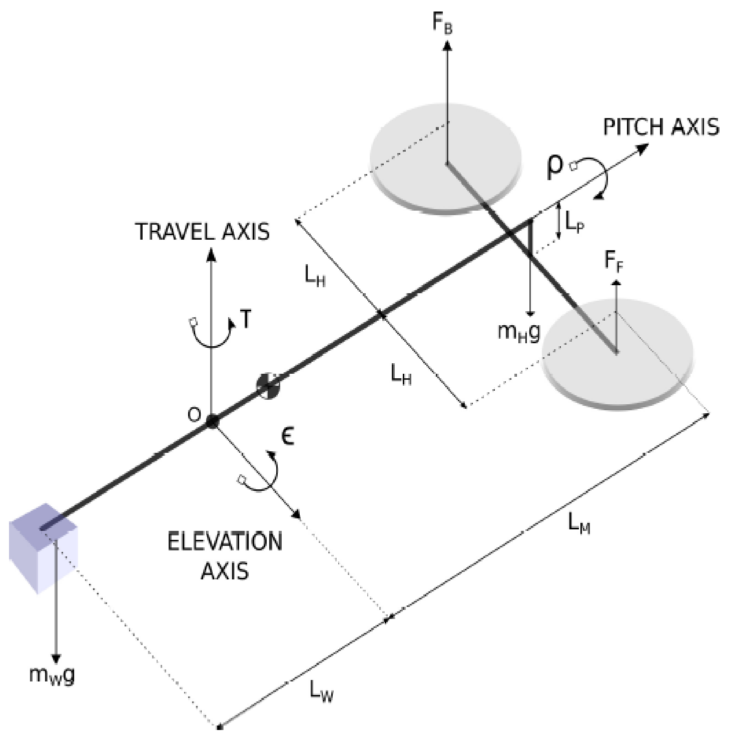


Figure 1.1. A basic schematic of the helicopter system[5]

1.3 Safety measures

In this process, the incorporation of a counterweight m_w serves the purpose of reducing the required power for achieving a consistent elevation in the helicopter's operation. The primary objective behind this utilization is to facilitate the validation and testing of the controller without imposing the necessity of dealing with weight constraints on the physical plant. This approach allows for subsequent adjustments in plant specifications, such as integrating motors with higher torque capabilities and employing batteries with more power, following successful controller validation. By employing this methodology, the focus remains on refining the controller's performance while ensuring flexibility for subsequent plant modifications to enhance its operational parameters. The controller can be easily transformed if different quantities are chosen since a hierarchical code in Matlab is designed to set the physical parameters prior to controller tuning.

1.4 Thrust identification

Starting with the thrust identification using an external bench test with the same propelling unit used in the helicopter, as shown in the figure 1.2. The aim is to find the resulting thrust for a given PWM voltage The thrust generated is manifested as a weight on the strain gauge, which is then interpreted as a thrust.

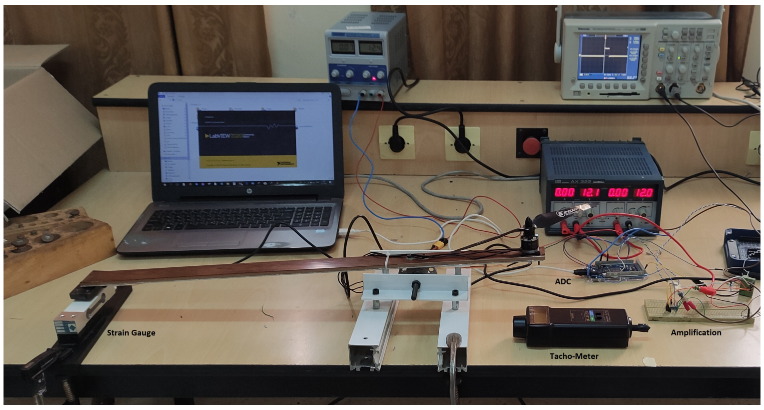


Figure 1.2. Stand for identifying the thrust unit

The values that were recorded are the applied PWM voltage, the RPM of the propeller, and the gauge voltage. This data is then mapped to the corresponding physical quantities by identifying the gauge. The data is then fitted to a static model due to the fact that its dynamics are much faster than the helicopter's.

$$F_{th} = 0.1623V_{pwm}^2 + 0.1678V_{pwm} + 0.02901 \tag{1.1}$$

1.5 Physical measurments

The physical quantities that could be directly measured on the plant such as the lengths were first conducted. This may not be the case for other quantities such as the counter which were incorporated when fitting the data from the different experiments. The following table encapsulates all of the directly measured quantities in table 1:

L_M	0.62 m
L_H	0.175 m
L_P	0.066 m

Table 1. Physically measured quantities

1.6 Modeling the three axes

Next, a full-body diagram of the system will be drawn, and from it, the equations of motion will be derived using Euler's second law of motion. This will be followed by identifying

the different parameters of the system (inertia, friction, masses...).
This chapter will conclude by transforming the equations describing the system into a state-space model that will be used in the controller design in the following chapter.

1.6.1 The pitch Axis

The free-body diagram of the helicopter around the pitch axis is shown in figure 1.3

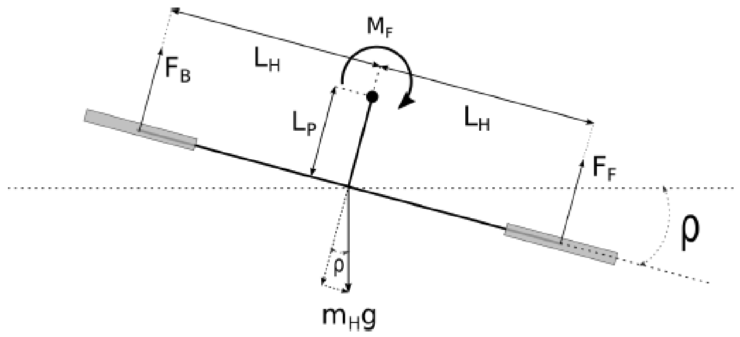


Figure 1.3. FBD of the pitch axis[5]

Parameters definitions:

- ρ: Pitch Angle
- L_P: The distance between the center of rotation of the pitch angle and the rig that carries the motors.
- L_H: The distance between the center of the propellers and the middle of the rig that carries the motors.
- m_H: The helicopter's front-end mass (including motors and ESCs etc).
- F_F: The thrust force of the front motor.
- F_B: The thrust force of the back motor.
- M_{Fp}: The momentum of the combined kinetic friction and air resistance of the pitch joint.
- μ_p: The friction coefficient of the pitch axis.
- I_p:The rotational inertia around the pitch axis.

Equation of motion:

From the FBD, it is clear that the thrust produced by the motors, more precisely the difference between the thrust of the back motor and the front motor, is what produces the

majority of the torque around the pitch axis. Although when $p \neq 0$, the gravitational force from the assembly will also produce a torque around the pitch axis.
Euler's second law yields:

$$I_p\ddot{\rho} = F_F L_H - F_B L_H - m_H g L_\rho \sin\rho \cos\epsilon - M_{Fp}$$

$$I_p\ddot{\rho} = (F_F - F_B) L_H - m_H g L_p \sin\rho \cos\epsilon - M_{Fp}$$

$$I_p\ddot{\rho} = F_{\text{diff}} L_H - m_H g L_p \sin\rho \cos\epsilon - \mu_p\dot{\rho} \tag{1.2}$$

Pitch parameters identification

The experiment started with fixing the helicopter at an elevation angle of $\epsilon = 0$.
At small pitch angles: $\sin(\rho) \approx \rho$.
Hence, the pitch equation of motion becomes:

$$I_p\ddot{\rho} = F_{\text{diff}} L_H - m_H g L_p \rho - \mu_p\dot{\rho} \tag{1.3}$$

The transfer function is:

$$\frac{L_H}{I_p s^2 + \mu_p s + m_H g L_p} \tag{1.4}$$

The helicopter is then stepped in the pitch axis by a thrust force $F_{\text{diff}} = 0.8196$ N. This was achieved by giving a thrust to the front motor $F_B = 1.77$ N (equivalent to 2.78V) and a thrust to the back motor $F_F = 0.9562$ N (equivalent to 1.78V). The response is then recorded by LabView and transferred to MATLAB. The following response in Figure 1.4 was obtained.

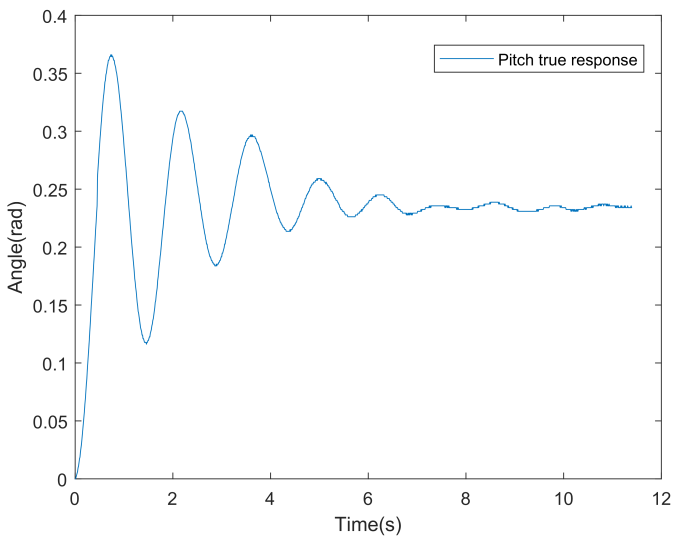


Figure 1.4. Recorded pitch response

The recorded response was exported to MATLAB, and the Matlab function **tfest**[6] was used to estimate the transfer function, the code can be found in Chapter one appendix. The following transfer function was obtained:

$$G_{p,est} = \frac{5.502}{s^2 + 0.9843s + 18.28} \tag{1.5}$$

The following plot demonstrates the comparison between the actual response and the simulated one using the obtained transfer function1.5:

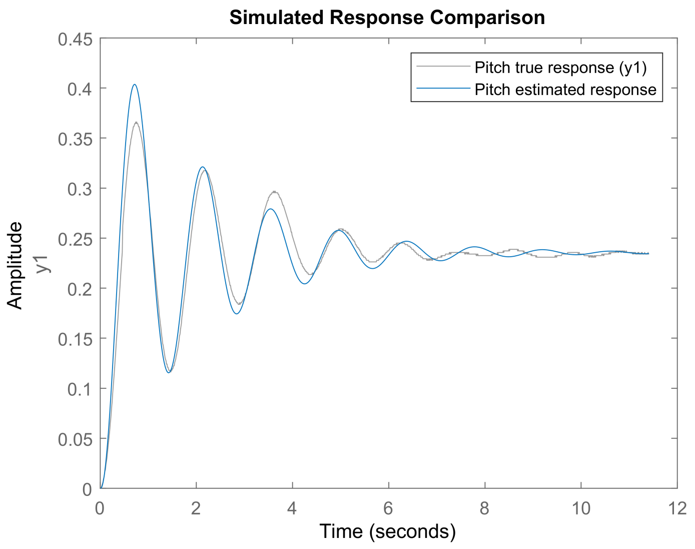


Figure 1.5. Comparison between simulated and experimental pitch response

The Mean Squared Error (MSE) is a commonly used metric to evaluate the performance of a predictive model or estimate the accuracy of predictions. It measures the average squared difference between the predicted values and the actual values in a dataset. The mse in this case is equal to 0.0157 rad.

The parameters were then obtained by matching the obtained transfer function 1.5 and the original transfer function 1.4 The estimated parameters are shown in the following table 2:

I_p	$0.0318 Kgm^2$
μ_p	$0.0313\ \frac{Nms}{rad}$
m_H	0.8978 kg

Table 2. Pitch equation identified parameters

1.6.2 Elevation Axis

The FBD of the helicopter around the elevation axis is demonstrated in Figure 1.6

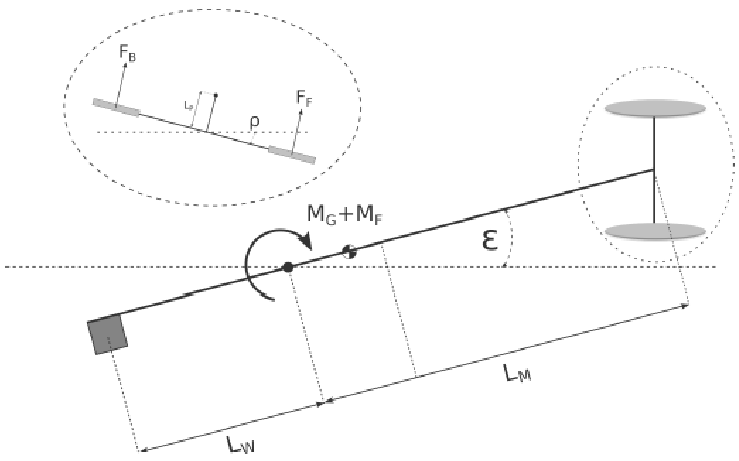


Figure 1.6. FBD of the Elevation axis[5]

Parameters definitions

- ϵ : Elevation Angle
- L_M: The distance between the center of elevation and the front end of the helicopter.

- L_w: The distance between the center of elevation and the center of gravity of the counterweight.
- M_W: The mass of the counterweight in the lower end of the rig.
- $M_G(\epsilon)$: The gravitational momentum of all the weight forces acting on the helicopter as a function of elevation angle
- $M_{F\epsilon}$: The momentum of the combined kinetic friction and air resistance of the elevation joint.
- μ_ϵ: The friction coefficient of the elevation axis.
- I_ϵ:The rotational inertia around the elevation axis.

Equation of motion

Euler's second law yields:

$$I_\epsilon \ddot{\epsilon} = F_F L_M \cos p + F_B L_M \cos p - M_G(\epsilon) - M_{F\epsilon}$$

$$I_\epsilon \ddot{\epsilon} = (F_F + F_B) L_M \cos p - M_G(\epsilon) - M_{F\epsilon}$$

$$I_\epsilon \ddot{\epsilon} = F_{\text{sum}} L_M \cos \rho - M_G(\epsilon) - \mu_\epsilon \dot{\epsilon} \tag{1.6}$$

Elevation parameters identification

Gravitational momentum

The main impeding force in the elevation of the 3DOF is gravity, which is opposed to the lifting force generated by the propellers. This suggests determining the thrust required to reach a particular elevation. Simply, the thrust is swept and held to reach a steady state elevation before recording the attained angle. This will be used in the next section to model the dynamics of the elevation axis. These angles are recorded from the elevation encoder in LabVIEW after mapping the PWM signal applied using the thrust equation, it is pointed out that this data is for one motor only. Since the nature of motion is rotational, we obtain the gravitational momentum equation by multiplying the fitted data by L_M

$$M_g(\epsilon) = -0.2200\epsilon^2 + 2.5278\epsilon + 1.7305 \tag{1.7}$$

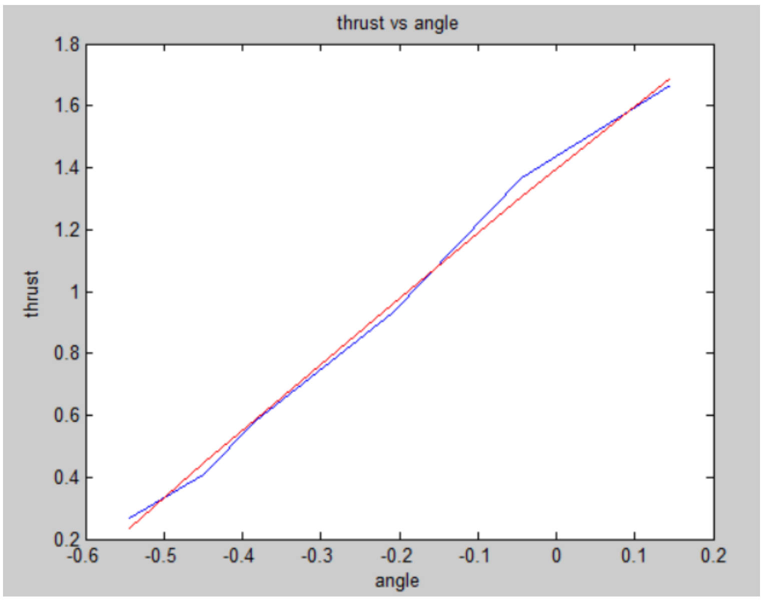


Figure 1.7. Thrust required at each elevation

Inertia and friction coefficient identification

The non-linear least squares method was used for the identification of these parameters. NLLS, or nonlinear least squares, is a potent method for estimating nonlinear model parameters that is applied in many different fields. It is especially helpful when linear models are insufficient because the connection between the model parameters and the observable data is not linear. The objective of NLLS is to reduce the squared difference between the predicted values of the model and the actual data points. This is accomplished by an iterative optimization procedure that updates the model parameters to reduce the residuals[7].

The fminunc function in MATLAB is frequently used to carry out NLLS optimization. The built-in optimization function fminunc, which stands for "function minimization unconstrained," is part of the MATLAB Optimization Toolbox. It is intended to determine the minimum of a multivariable function that is not restricted [8].
For using the fminunc, an objective function must be defined. This objective function outputs the sum of squares of the residuals and accepts the model parameters as inputs. After that, the fminunc function executes iterations, modifying the parameters to minimize the objective function up to convergence or the fulfillment of a predetermined termination criterion.
The elevation system was fed with an input thrust of approximately $F_{\text{sum}} = 2.5407\text{N}$ (equivalent to 2.3V PWM to each motor), this made the helicopter hover at an angle of -2 degrees. The system identification process then started, and the elevation was fed with different inputs. The response was recorded using Labview and then transferred to Matlab to identify the parameters.
The following plot shows the elevation response and the inputs used:

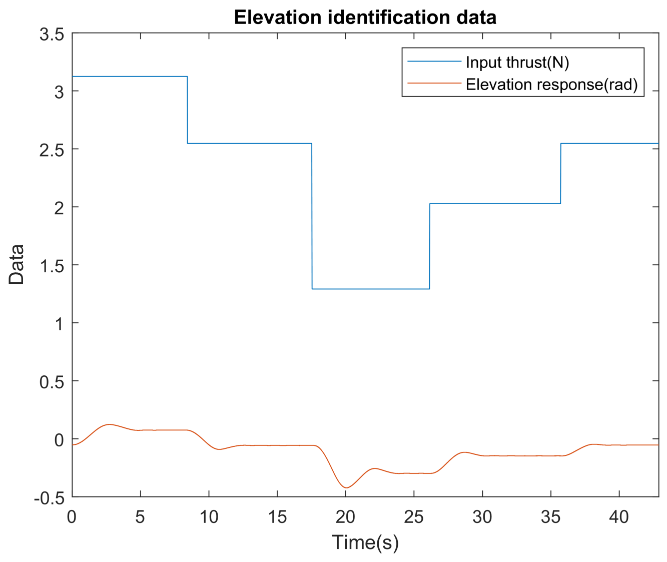


Figure 1.8. Recorded elevation response and the inputs used

The Matlab codes used for identification and for simulating the non-linear model are found in the chapter one appendix. The following parameters were obtained and are recorded in the following table 3:

I_ϵ	$1.3434Kgm^2$
μ_ϵ	$1.5476\frac{Nms}{rad}$

Table 3. Elevation equation identified parameters

The following plot 1.9 shows a comparison between the simulation using the obtained parameters and the true response of the system.

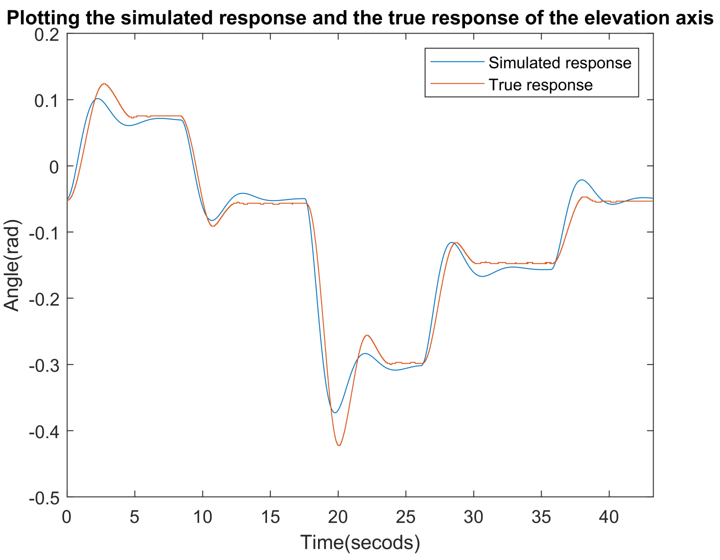


Figure 1.9. Experimental elevation data vs simulated model response

1.6.3 Travel Axis

The FBD of the helicopter around the elevation axis is demonstrated in Figure 1.10:

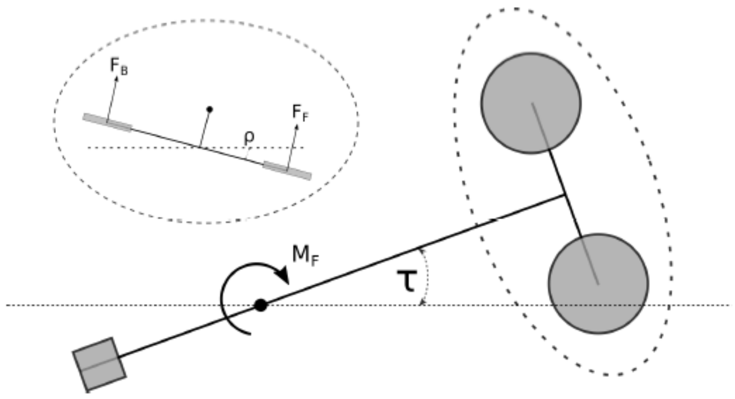


Figure 1.10. FBD of the travel axis [5]

Parameters definitions

- τ : Travel angle.
- I_τ: Travel axis inertia.
- $M_{F\tau}$: The momentum of the combined kinetic friction and air resistance of the travel joint.

Equation of motion

The thrust produced by the two motors is the primary force operating on the travel axis. When the pitch angle is not zero, a component of the thrust forces will produce a torque around the travel axis. With a higher elevation angle, the component will both rise and decrease.

The travel inertia will fluctuate quite a bit when the helicopter flies around, in contrast to the pitch and elevation inertia.

It shall be stated as follows to demonstrate that the travel inertia is not constant but rather a function of the elevation angle: $I_\tau = I_\tau(\epsilon)$. Euler's second law yields:

$$I_\tau(\epsilon)\ddot{\tau} = (F_F + F_B)L_M \cos\epsilon \sin\rho - M_{F\tau}$$

$$I_\tau(\epsilon)\ddot{\tau} = F_{\text{sum}}L_M \cos\epsilon \sin\rho - \mu_\tau \dot{\tau} \tag{1.8}$$

Travel parameters identification

The identification of the travel axis was a little bit challenging, as can be seen from its equation of motion, which is highly affected by both pitch and elevation.

For the travel identification, the linear state-space equation of motion of the travel axis 1.9 that will be obtained and discussed further in the following chapter, is used to identify the travel axis parameters.

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_5} \\ \dot{x_6} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{-m_H g L_P}{I_p} & -\frac{\mu_p}{I_p} & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{M(0)}{I_\tau(0)} & 0 & 0 & -\frac{u_\tau}{I_\tau(0)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_5 \\ x_6 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{L_H}{I_p} \\ 0 \\ 0 \end{bmatrix} U_{\text{diff}}$$

$$y = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_5 \\ x_6 \end{bmatrix} \tag{1.9}$$

This state space equation is the result of linearizing the travel system around the 0 elevation angle. This equation will be used because, in this project, the travel system will be linearized and controlled around an elevation angle of 0 so the travel behavior around this region is the main topic of interest. the procedure of identification was as follows:

First, the elevation angle is fixed at angle 0, this is done by applying an F_{sum} thrust equals to 2.73N (2.4V PWM to each motor).Then an F_{diff} thrust of 0.8196N is applied. After 2 seconds, F_{diff} is set back to 0.

The whole response is recorded and then transferred to Matlab to process it.

The following plot 1.11 was obtained:

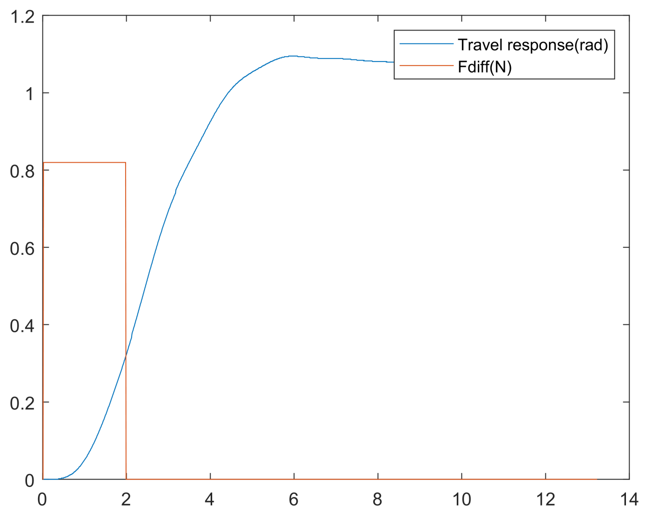


Figure 1.11. The recorded travel response and the input used

The linear least squares method is used for the identification. The travel-pitch state space equation has only two unknown parameters: the travel friction coefficient μ_τ and the travel inertia at elevation 0 $I_\tau(0)$. The other parameters have been estimated in the pitch and elevation identifications. The codes used for identification can be found in Chapter 1 of the appendix.
The identification gave the following results in table 4 :

$I_\tau(0)$	$1.1464 Kgm^2$
μ_τ	$0.6366 \frac{Nms}{rad}$

Table 4. Travel identified parameters

The following figure shows a comparison between the true and simulated responses:

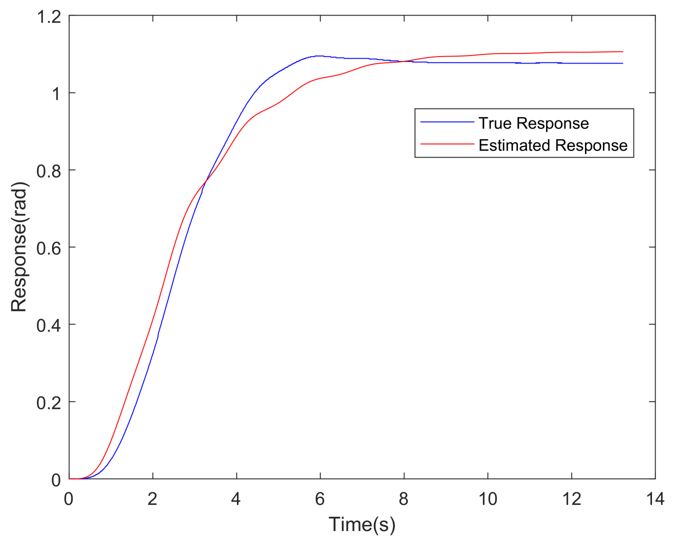


Figure 1.12. Actual response versus the simulated one.

The mean square error is: 0.0087rad.

Multiple inputs were fed to the system to validate the model, the following plot 1.13 was obtained:

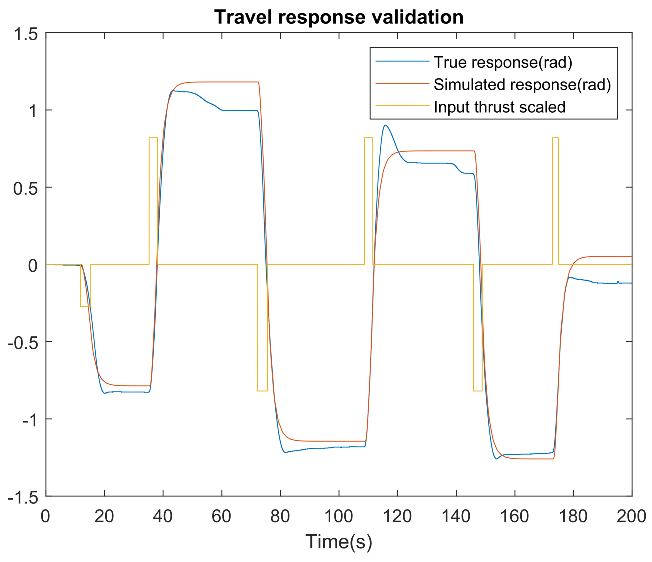


Figure 1.13. Travel model validation

The mean square error is: 0.01rad

1.7 State-space model derivation

After the parameters were identified one can now proceed and develop the state-space model.
The system is a nonlinear system, so the state-space model is on the form:

$$\dot{x} = f(x, u)$$
$$y = h(x)$$

Where x denotes the state vector, u is the input vector, and y is the output vector. the state variables are the angles and their corresponding angular velocities:

$$\begin{aligned} x_1 &= \rho \\ x_2 &= \dot{\rho} \\ x_3 &= \epsilon \\ x_4 &= \dot{\epsilon} \\ x_5 &= \tau \\ x_6 &= \dot{\tau} \end{aligned}$$

the following state-space equation is obtained:

$$\begin{aligned} \dot{x_1} &= x_2 \\ \dot{x_2} &= \frac{U_{\text{diff}} L_H - m_H g L_P \sin x_1 \cos x_3 - \mu_p x_2}{I_p} \\ \dot{x_3} &= x_4 \\ \dot{x_4} &= \frac{U_{\text{sum}} L_M cos x_1 - M(x_3) - \mu_\epsilon x_4}{I_\epsilon} \\ \dot{x_5} &= x_6 \\ \dot{x_6} &= \frac{U_{\text{sum}} L_M cos x_3 sin x_1 - \mu_\tau x_6}{I_\tau(x_3)} \end{aligned} \tag{1.10}$$

Where :

$$\begin{aligned} U_{\text{sum}} &= F_F + F_B \\ U_{\text{diff}} &= F_F - F_B \end{aligned} \tag{1.11}$$

1.8 Conclusion:

In this chapter, the equations of motion have been developed using Euler equations, and the parameters have been identified finally a nonlinear state space model has been derived which will be used in the following chapter for designing a controller.

Chapter 2. Controller design

After modeling the system and obtaining the necessary parameters in the previous chapter, a controller for the system will be designed in this chapter.

2.1 System linearization

In this thesis, the controllers used are linear. and since the system of interest is a nonlinear system, it must be linearized. The linearization process starts by calculating the equilibrium point. and then linearize the system around this equilibrium point using the Jacobian method.

2.1.1 Calculating the equilibrium point

A point where the rate of change of all the states is zero is called an equilibrium point or in other words, the system remains there forever when it reaches it. in order to calculate the equilibrium point, we will solve for the equation $f(\overline{x}, \overline{u}) = 0$:

$$\begin{aligned}
0 &= \overline{x_2} \\
0 &= \frac{U_{\text{diff}} L_H - m_H g L_P \sin \overline{x_1} \cos \overline{x_3} - \mu_p \overline{x_2}}{I_p} \\
0 &= \overline{x_4} \\
0 &= \frac{U_{\text{sum}} L_M cos \overline{x_1} - M(\overline{x_3}) - \mu_\epsilon \overline{x_4}}{I_\epsilon} \\
0 &= \overline{x_6} \\
0 &= \frac{U_{\text{sum}} L_M cos \overline{x_3} sin \overline{x_1} - \mu_\tau \overline{x_6}}{I_\tau(x_3)}
\end{aligned} \tag{2.1}$$

The system of equations 2.1 gives $\overline{x_2} = \overline{x_4} = \overline{x_6} = 0$ and :

$$\begin{aligned}
0 &= \frac{U_{\text{diff}} L_H - m_H g L_P \sin \overline{x_1} \cos \overline{x_3}}{I_p} \\
0 &= \frac{U_{\text{sum}} L_M cos \overline{x_1} - M(\overline{x_3})}{I_\epsilon} \\
0 &= \frac{U_{\text{sum}} L_M cos \overline{x_3} sin \overline{x_1}}{I_\tau(x_3)}
\end{aligned} \tag{2.2}$$

Three equations and four unknowns make up this system, which means that there are an infinite number of equilibrium points. One of the unknowns in this system needs to be fixed in order to solve it.

It can be shown that there are three ways to solve the third equation in the system by analyzing it. One may select U_{sum} = 0, $\overline{x_1} = 0$, or $\overline{x_3} = \frac{\pi}{2}$. Here, choosing $\overline{x_1} = 0$ is the relevant option.this results in:

$$
\begin{aligned}
0 &= \frac{U_{\text{diff}} L_H}{I_p} \\
0 &= \frac{U_{\text{sum}} L_M - M(\overline{x_3})}{I_\epsilon}
\end{aligned}
\tag{2.3}
$$

The first equation in the system 2.3 gives $U_{\text{diff}} = 0$ The second equation gives the thrust at an elevation angle, choosing the state for the linearization to be $\overline{x_3 = 0}$, get:

$$
U_{\text{sum}} = \frac{M(0)}{L_M} \tag{2.4}
$$

Note that none of the aforementioned equations include the state x_5 (travel angle), indicating that its value has no bearing whatsoever on the resulting state space.

2.1.2 Obtaining the linearized model

After selecting the equilibrium points, the Jacobian linearization method around the equilibrium point will be used to obtain the linearized state space equation, which resulted in :

$$
A = \frac{\partial f}{\partial x}(\overline{x}, \overline{u}) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{-m_H g L_P}{I_p} & -\frac{\mu_P}{I_p} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{M'(0)}{I_\epsilon} & -\frac{\mu_\epsilon}{I_\epsilon} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \frac{M(0)}{I_\tau(0)} & 0 & 0 & 0 & 0 & -\frac{u_\tau}{I_\tau(0)} \end{bmatrix} \tag{2.5}
$$

$$
B = \frac{\partial f}{\partial u}(\overline{x}, \overline{u}) = \begin{bmatrix} 0 & 0 \\ 0 & \frac{L_H}{I_p} \\ 0 & 0 \\ \frac{L_M}{I_\epsilon} & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{2.6}
$$

The outputs of the system are the elevation and travel angles. Hence, The C and D matrices will be chosen as follows:

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.7}$$

$$D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{2.8}$$

Where A is the system matrix, B is the input matrix, C is the output matrix, and D is the feedforward matrix.
The linearized state-space model is:

$$\begin{gathered}
\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \\ \dot{x_5} \\ \dot{x_6} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{-m_H g L_P}{I_p} & -\frac{\mu_p}{I_p} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ -\frac{M'(0)}{I_\epsilon} & -\frac{\mu_\epsilon}{I_\epsilon} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \frac{M(0)}{I_\tau(0)} & 0 & 0 & -\frac{u_\tau}{I_\tau(0)} & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & \frac{L_H}{I_p} \\ 0 & 0 \\ \frac{L_M}{I_\epsilon} & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} U_{\text{sum}} \\ U_{\text{diff}} \end{bmatrix} \\
y = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix}
\end{gathered} \tag{2.9}$$

2.2 System characteristics

2.2.1 Decoupling the system

One of the techniques to create a controller for a Multiple Input Multiple Output (MIMO) system is to decouple it by matching inputs and outputs to create two distinct systems that work independently of one another (Single Input Single Output, or SISO). Only when the interactions between the selected pairings are minimal is the system appropriate for decoupling. it can be seen from the linearized matrix A that the system can be decoupled into two systems: the elevation angle which only responds to the U_{sum} input and the travel angle which only responds to the U_{diff} input, we can conclude that the elevation and travel systems are entirely independent. Hence, we may split our system into two SISO systems by decoupling it from a MIMO system.

Elevation system

$$\begin{gathered}\begin{bmatrix}\dot{x}_3\\ \dot{x}_4\end{bmatrix} = \begin{bmatrix}1 & 0\\ -\frac{M'(0)}{I_\epsilon} & -\frac{\mu_\epsilon}{I_\epsilon}\end{bmatrix}\begin{bmatrix}x_3\\ x_4\end{bmatrix} + \begin{bmatrix}0\\ \frac{L_M}{I_\epsilon}\end{bmatrix} U_{\text{sum}}\\ y = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}x_3\\ x_4\end{bmatrix}\end{gathered} \tag{2.10}$$

Travel system:

$$\begin{gathered}\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_5\\ \dot{x}_6\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0\\ \frac{-m_H g L_P}{I_p} & -\frac{\mu_p}{I_p} & 0 & 0\\ 0 & 0 & 0 & 1\\ \frac{M(0)}{I_\tau(0)} & 0 & 0 & -\frac{u_\tau}{I_\tau(0)}\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_5\\ x_6\end{bmatrix} + \begin{bmatrix}0\\ \frac{L_H}{I_p}\\ 0\\ 0\end{bmatrix} U_{\text{diff}}\\ y = \begin{bmatrix}0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_5\\ x_6\end{bmatrix}\end{gathered} \tag{2.11}$$

2.2.2 Stability analysis

The behavior of a system following a deviation from its ideal condition of operation is the subject of stability.

- The location of the eigenvalues of the matrix A determines the stability (asymptotic) of a system defined by a state equation: The system is considered stable if all eigenvalues, Re$\{\lambda_i\}$, have negative real parts. On the other hand, if at least one eigenvalue has a positive real part, the system is unstable. In the case where there are non-repeated eigenvalues lying on the $j\omega$ axis, the system exhibits marginal stability. Otherwise, the system is deemed unstable.
- For a system described by a transfer function, stability (BIBO, bounded-input bounded-output) is determined by the location of the poles in a similar manner.

Elevation system stability analysis

The matrix A of the elevation system is:

$$A = \begin{bmatrix} 1 & 0 \\ -\frac{M'(0)}{I_\epsilon} & -\frac{\mu_\epsilon}{I_\epsilon} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -1.8816 & -1.1520 \end{bmatrix}$$

The eigenvalues are: -0.5760 + 1.2449i,-0.5760 - 1.2449i
Since the real part of the eigenvalues is less than 0, the elevation system is asymptotically stable.

Travel system stability analysis

The matrix A of the elevation system is:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{-m_H g L_P}{I_p} & -\frac{\mu_p}{I_p} & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{M(0)}{I_\tau(0)} & 0 & 0 & -\frac{u_\tau}{I_\tau(0)} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -18.28 & -0.9843 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1.507 & 0 & 0 & -0.55 \end{bmatrix}$$

The eigenvalues are: 0,-0.5500,-0.4921 + 4.2470i,-0.4921 - 4.2470i
Since there is an eigenvalue equal to 0, this indicates that the travel system is marginally stable.

2.2.3 Controllability analysis

One of the key ideas in control theory is controllability. It describes the capacity of external input to change the system's state from any initial state to any final state in a finite amount of time. A system is deemed controllable if and only if its controllability matrix is full rank. The controllability matrix is defined as:

$$C_o = [B\ AB\ A^2\ B...\ A^{n-1}B]$$

Where A and B are the state and input matrices of the system, respectively.
The controllability matrix can be computed using the ctrb Matlab command.

Elevation system Controllability analysis

The matrix B for the elevation system is:

$$B = \begin{bmatrix} 0 \\ 0.4615 \end{bmatrix}$$

Using the Matlab **ctrb** command, the controllability matrix is :

$$C_o = ctrb(A, B) = \begin{bmatrix} 0 & 0.4615 \\ 0.4615 & -0.5316 \end{bmatrix}$$

The determinant of the controllability matrix Co:

$$det(C_o) = -0.2130 \neq 0$$

It implies that the controllability matrix C_o is full rank. Hence, the elevation system is fully controllable.

Travel system Controllability analysis

The matrix B for the travel system is:

$$B = \begin{bmatrix} 0 \\ 5.503 \\ 0 \\ 0 \end{bmatrix}$$

The controllability matrix is :

$$C_o = \begin{bmatrix} 0 & 5.5031 & -5.4166 & -95.2635 \\ 5.5031 & -5.4166 & -95.2635 & 192.7790 \\ 0 & 0 & 0 & 8.2944 \\ 0 & 0 & 8.2944 & -12.7259 \end{bmatrix}$$

The determinant of the controllability matrix Co:

$$det(C_o) = -2.0835 \times 10^3 \neq 0$$

Which implies that the travel system is fully controllable.

2.2.4 Observability analysis

A further key idea in control theory is observability. It refers to the capacity of an external output to ascertain a system's internal state. A system is deemed observable if and only if its observability matrix is full rank.

The observability matrix is defined as:

$$O_b = [C; CA; CA^2; ...; CA^{n-1}]$$

Where A and C are the state and output matrices of the system, respectively. The observability matrix can be computed using the obsv Matlab command.

Elevation system Observability analysis

The matrix C for the elevation system is:

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

Using the Matlab **obsv** command, the observability matrix is:

$$O_b = obsv(A, B) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The determinant of the observability matrix O_b:

$$det(O_b) = 1 \neq 0$$

It implies that the observability matrix O_b is full rank. Hence, the elevation system is fully observable.

Travel system Observability analysis

The matrix C for the travel system is:

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

The observability matrix is:

$$O_b = obsv(A, B) = \begin{bmatrix} 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 1.0000 \\ 1.5072 & 0 & 0 & -0.5500 \\ -0.8290 & 1.5072 & 0 & 0.3025 \end{bmatrix}$$

The determinant of the observability matrix O_b:

$$det(O_b) = 2.2717 \neq 0$$

Hence the travel system is fully observable.

2.3 Nonlinearities compensation using Feedback linearization for the elevation system

In this project, the controllers used are linear controllers. The controller for the elevation system is desired to operate at the full range of angles and not only around the 0 operating point .Simply linearizing it using the Jacobian method will not be sufficient as there are some non nonlinearities that cannot be neglected, figure 2.1 shows a comparison between linearized elevation system using the Jacobian method 2.10 and nonlinearized system responses to the same input, the simulation code used for simulation can be found in chapter 2 of the Appendix.

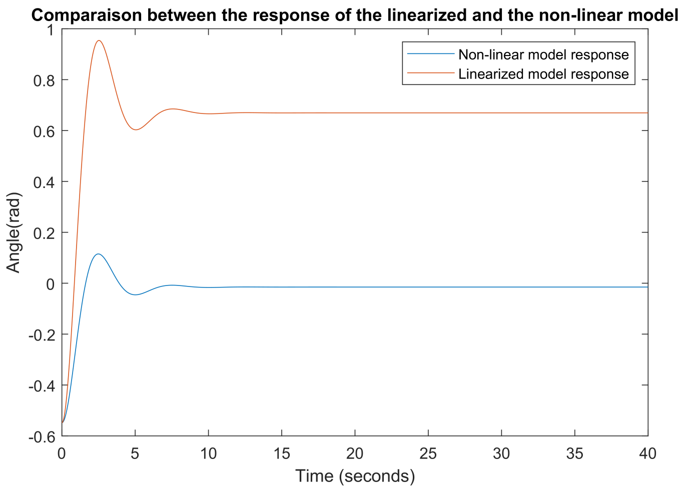


Figure 2.1. The linear and non-linear elevation state space response to the same input.

To compensate for the nonlinearities a technique called feedback linearization will be used. By using the proper feedback signals, feedback linearization is a control method used to linearize a nonlinear system.
In order to use linear control techniques, the approach’s main idea is to algebraically convert nonlinear system dynamics into a (fully or partially) linear one. This completely varies from traditional linearization techniques like Jacobian linearization in that, feedback linearization is accomplished by precise state modifications and feedback rather than through linear dynamics approximations.

2.3.1 Feedback linearization for systems in the companion form

A class of nonlinear systems that are characterized by the so-called companion form, or controllability canonical form, can easily be subjected to feedback linearization, which involves canceling the nonlinearities and imposing desired linear dynamics[9]. A system is considered to be in companion form if the equation below accurately describes its dynamics:

$$x^{(n)} = f(x) + b(x)u$$

Where f(x) and b(x) are nonlinear functions.
The state space representation of the above equation is:

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ \dots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} x_2 \\ \dots \\ x_n \\ f(x) + b(x)u \end{bmatrix}$$

For this system use the control input:

$$u = \frac{1}{b}[v - f]$$

The nonlinearities can be canceled, and a simple input-output relation is obtained:

$$x^{(n)} = v$$

As a result, the dynamics of the system take on a linear form, which allows for analysis and control design utilizing linear control methods such as PID control, state feedback, or LQR (Linear Quadratic Regulator).

2.3.2 Compensating the nonlinearities in the elevation system:

If the pitch angle is small enough, $\cos p \approx 1$ the non-linear state space for the elevation system becomes:

$$\dot{x_3} = x_4$$
$$\dot{x_4} = \frac{U_{\text{sum}} L_M - (ax_3^2 + bx_3 + c) - \mu_\epsilon x_4}{I_\epsilon}$$

Where $ax_3^2 + bx_3 + c$ is the equation of the gravitational momentum.

The elevation system is in the companion form.
The goal is to choose the input U_{sum} that cancels out the terms $-ax_3^2 - c$ If the input is chosen to be :

$$U_{sum} = v + \frac{ax_3^2 + c}{L_M}$$

The elevation state space becomes:

$$\dot{x_3} = x_4$$
$$\dot{x_4} = \frac{vL_M - bx_3 - \mu_\epsilon x_4}{I_\epsilon}$$

v is the input generated from the linear controller like LQR, and the linear system generated above 2.10 using the Jacobian method is used for the tuning. Figure 2.2 shows the result of applying the same input after using feedback linearization. To use the above feedback linearization design when the nonlinear dynamics are not in a controllability canonical form, one may need to first use algebraic transformations to convert the dynamics into the controllability form, or one may need to rely on partial linearization of the original dynamics rather than full linearization[9].

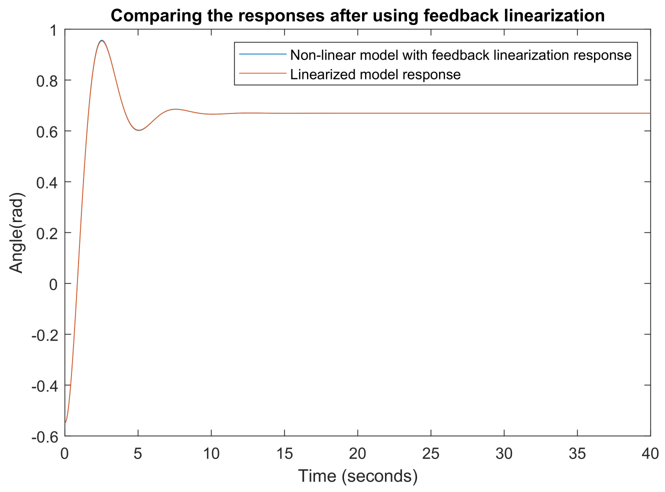


Figure 2.2. The linear and non-linear elevation state space response to the same input after using feedback linearization.

2.4 State feedback controllers

A state feedback controller is a type of controller that determines the control signal based on the complete state of the system. The selection of individual feedback gains for the whole collection of state variables is referred to as the state feedback controller design. All of the state variables are thought to be accessible for observation. The system's transient response and steady-state tracking error reduction are the design objectives. Figure 2.3 shows a basic schematic for the state feedback controller, so the goal is to find the gain K that gives the desired response.

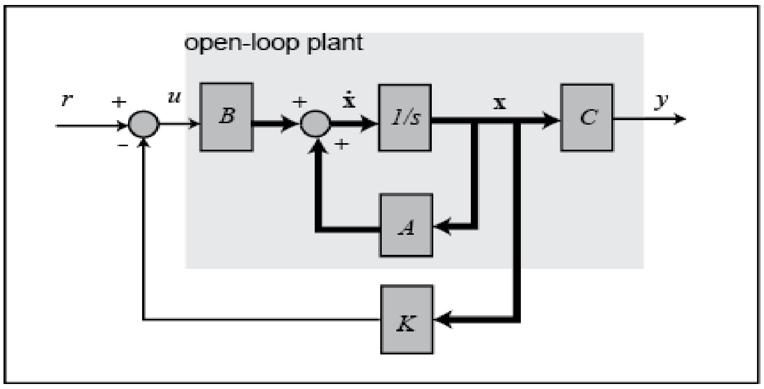


Figure 2.3. a basic schematic of the state feedback controller.[10]

There are two common methods to calculate the gain K: the Pole placement method and the Linear quadratic regulator(LQR) method.

2.4.1 Pole placement

Pole placement, which involves defining the intended closed-loop poles location, is a common technique for developing state feedback controllers. The closed-loop poles determine the system's response properties, such as stability, transient response, and damping. pole placement involves two steps:

Poles selection

The closed-loop pole locations are the eigenvalues of A-BK. The closed-loop system's desired poles should be chosen based on the desired system response, The settling time, overshoot, and damping of the system will all be determined by the pole locations. The chosen pole locations must satisfy the intended control goals while guaranteeing stability.

Controller design:

This step involves calculating the gain K with the intended pole sites in mind so that the closed-loop system achieves the pole locations. By resolving a series of linear algebraic equations generated from the system's model and desired pole locations, the gain vector can be found. The particular method for resolving these equations depends on the features

of the system and the methodology for placing the poles that have been selected. a simple way is to use the **place** in Matlab that assigns the poles to any desired locations in the complex plane.
It is crucial to remember that the pole placement method assumes full-state feedback, which means that the system's entire state vector is available for feedback. When the complete state is not directly measurable, state estimation techniques like observers or Kalman filter may be used in practice.
The pole placement method gives designers more freedom when creating control systems that adhere to particular performance standards. However, it is essential to make sure that the chosen pole positions do not go against the rules for system stability. To create a robust and stable closed-loop system, careful analysis and consideration should be given to the system dynamics, stability margins, and potential interactions with other control loops or disturbances.
Due to its inherent benefits, the Linear Quadratic Regulator (LQR) is often chosen over pole placement when building state feedback controllers. While the direct specification of desirable closed-loop pole positions is made possible by pole placement, LQR goes above and beyond by offering the best possible control strategy. The system dynamics, control effort, and predetermined performance criteria are all taken into account by LQR when forming the control issue as an optimization task. It allows engineers to establish a fair trade-off between many objectives, such as settling time, control effort, and state errors, by optimizing a quadratic cost function. LQR also has the ability to manage constraints on control inputs and states, gives resilience against disturbances and uncertainties, and allows for simple tweaking of controller behavior using adjustable weighting factors.
LQR is a strong and adaptable technique for creating efficient state feedback controllers that offer improved performance and resilience because it can maximize performance, take uncertainties into account, and solve restrictions.

2.4.2 LQR controller

The Linear quadratic regulator (LQR) is a mathematical approach used in control theory to identify the best control rule for a linear system with a quadratic cost function. The LQR is probably the most significant result in optimal control theory to date. The LQR method seeks to optimize a quadratic cost function that encapsulates the trade-off between control effort and system performance. It offers a methodical way to balance conflicting goals and attain optimal control.

LQR design procedure

Let the model be represented by a set of state-space equations with the notation are as stated previously:

$$\dot{X} = AX + Bu$$

$$y = CX + Du$$

The performance criteria must be determined that describe the desired optimization. Typically, this involves specifying a cost function that represents a trade-off between control effort and system performance.
State errors, control inputs, and their rates of change are usually included in the cost function. Each objective's importance is determined by the relative weights that have been given to these terms. The general form of the LQR cost function is given by:

$$J = \int (X^T QX + u^T Ru)dt$$

X is the state vector of the system, u is the control input vector, Q is the state weight matrix, and R is the control effort weight matrix.
The penalty for state errors is determined by the state weight matrix **Q**, which is a symmetric positive semidefinite matrix $(X^T QX \geq 0)$. A higher cost or penalties for deviations in the related state variables is indicated by higher values in the diagonal elements of Q. Various states with varying degrees of importance can be assigned based on the desired control objectives by modifying the values in Q.
The penalties on control inputs are represented by the control effort weight matrix **R**, which is symmetric positive definite $((u^T Ru > 0))$. Higher values in the diagonal elements of R, similar to Q, denote a higher cost or penalty for more control efforts. This enables engineers to modify the settings in R to regulate how aggressive the controller is. The controller can prioritize energy economy or reduce the wear and tear on actuators by stiffening the penalties for excessive control effort.
Finding the optimal state feedback gain matrix K with the smallest cost function J is the primary goal of the LQR design. The LQR algorithm computes the optimal gain matrix to achieve a trade-off between state-tracking performance and control effort by choosing suitable values for Q and R. this is achieved by solving the associated Riccati equation:

$$AS^T + SA + SBR^{-1}B^T S + Q = 0$$

the solution S is then used to compute the optimal gain K by the equation

$$K = R^{-1}B^TS$$

The Riccati equation's solution guarantees that the resulting controller minimizes the required cost function.
Once the optimal control gain vector K is calculated, the LQR controller can be implemented by applying the control law $u = -KX$ to the system as shown in Figure 2.3. The controller calculates the control input based on the current state of the system.

Reference tracking

After calculating the optimal gain K, the output y is set to track a reference trajectory r. there are two ways to achieve that:
LQR with forward gain: As shown in figure 2.4, To achieve reference tracking, the gain $\bar{N}$ must be carefully selected.

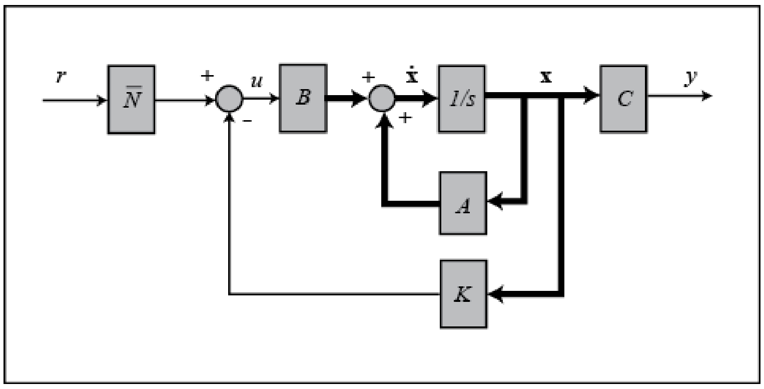


Figure 2.4. LQR with forward gain[10]

It is evident that :

$$\dot{X}(t) = (A + BK)X(t) + BNr(t)$$

$$y(t) = CX(t)$$

To have $y(t) \longrightarrow r$, a DC gain from $r \longrightarrow y(t)$ must obtained:

$$C(I - (A + BK))^{-1}BN = I$$

Hence, $\bar{N}$ is calculated using the following equation:

$$\bar{N} = (C(I - (A + BK))^{-1}B)^{-1}[11]$$

Despite the advantages of the forward gain technique, there are certain drawbacks and restrictions when utilizing simply a forward gain:

- Lack of Robustness: The forward gain strategy may be less resistant to unforeseen events and changes in the system dynamics. It may not respond well to changing conditions or modeling errors because it only uses a fixed gain matrix. Any departure from the established system model may result in unstable or poor performance.
- Limited Compensation for Disturbances: The main goal of a forward gain is to influence how the system reacts to a reference input. It might not be able to fully make up for system output disturbances. Stable-state errors or performance decline might result from disturbances that are not taken into account during design.
- Achieving Precise Tracking May Be Difficult: The forward gain method by itself may have trouble achieving precise tracking of reference signals, especially when there are disturbances or model inaccuracies.
- Sensitivity to Model Mismatch: Since the forward gain is estimated using a specific system model, a mismatch between the model's presumptions and the behavior of the actual system may affect control effectiveness. The controller's ability to perform well in practical situations may be constrained by its sensitivity to model mismatches.

Most of these problems will be solved by using the integral action method.
LQR with integral action:
As stated earlier, a controller with just a forward gain will not be able to achieve a steady-state tracking error of zero because of model imperfections, external disturbances, and similar phenomena. To reduce this, integral action must be added to the feedback loop as shown in figure2.5.

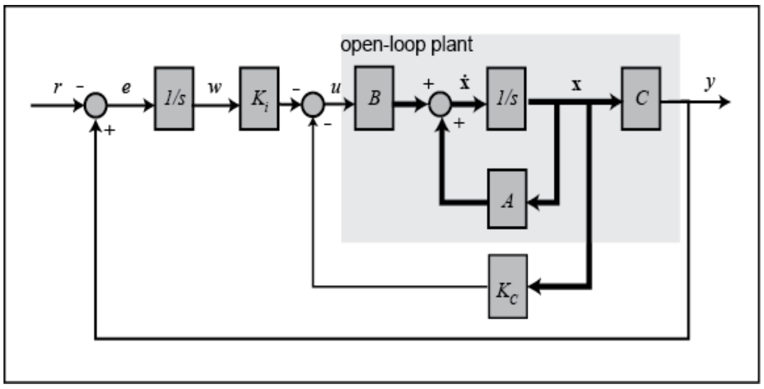


Figure 2.5. LQR with integral action[10]

By adding the integral of the error to the set of state variables, an integral action has been included in the LQR controller[12]. The system's space model gains one new state. The tracking error is defined as:

$$e = \dot{w} = r(t) - y(t)$$

After augmenting the integral of the error state, the system state space becomes:

$$\begin{aligned}\dot{\bar{X}} &= \bar{A}\bar{X} + \bar{B}\bar{u} \\ y &= \bar{C}\bar{X}\end{aligned}$$

Where:

$$\dot{\bar{X}} = \begin{bmatrix} \dot{X} \\ e \end{bmatrix}, \bar{A} = \begin{bmatrix} A & 0 \\ -C & 0 \end{bmatrix}, \bar{X} = \begin{bmatrix} X \\ w \end{bmatrix},$$
$$\bar{B} = \begin{bmatrix} B \\ 0 \end{bmatrix}, \bar{C} = \begin{bmatrix} C & 0 \end{bmatrix}$$

The state feedback optimal controller gain Kc and the integral gain Ki are calculated using the new augmented matrices.
LQR's capacity to simultaneously analyze several control objectives and strike an ideal balance between them is one of its primary benefits. It offers a methodical and reliable control design technique to manage both single-input single-output (SISO) and multi-input multi-output (MIMO) systems. LQR is particularly useful for the performance optimization of linear time-invariant systems.

2.5 LQR tunning

The LQR was tuned in Matlab. Since the controllers will be implemented digitally, the MATLAB function **lqrd**[13] was used for tuning. This function is used to discretize the continuous-time LQR. It takes the matrices of the continuous-time state-space model, the weighting matrices Q and R, and the sampling time. It uses the zero-order hold (ZOH) discretization method.
The sampling time is equal to 10ms, the reason for choosing it will be discussed in the following chapter.
The LQR can also be tuned in continuous time directly and the controller will still work perfectly fine when implemented digitally using 10ms sampling time. This is true according to the theorem that states: The implications of utilizing a discrete controller instead of a continuous one can be disregarded if the sampling frequency is at least 10 times the closed loop system's bandwidth[14]. From the bode plots of the closed-loop elevation and travel systems in figures 2.6 and 2.7 respectively, the elevation and travel closed-loop bandwidths are 0.278 Hz and 0.1277 Hz respectively. The sampling frequency in our case is equal

to 100 Hz which is much higher than 10 times the sampling frequency of both of them. Hence, it is possible to ignore the fact that the controller is discrete.

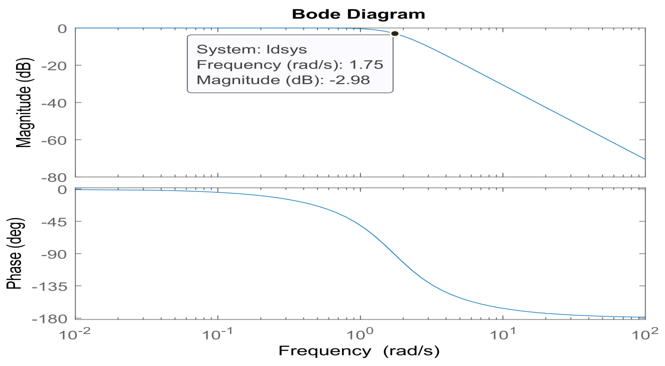


Figure 2.6. Bode plot of the closed-loop elevation system.

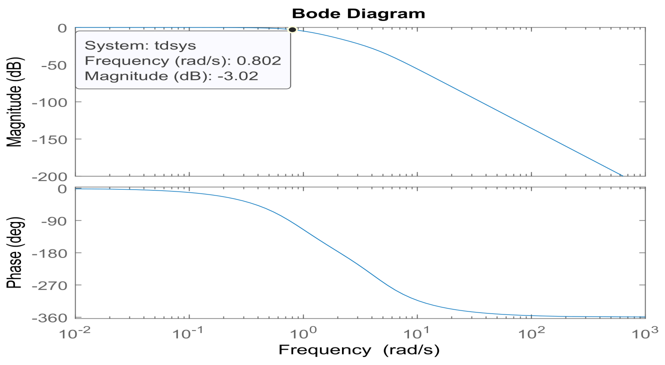


Figure 2.7. Bode plot closed-loop travel system.

2.5.1 Tuning the LQR for the elevation

The LQR with an integral action is used.
Augmenting the elevation system with the error state described earlier gives the following state-space model:

$$\begin{bmatrix} \dot{x}_3 \\ \dot{x}_4 \\ e \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -\frac{M'(0)}{I_\epsilon} & 0 & -\frac{\mu_\epsilon}{I_\epsilon} \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \\ w \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{L_M}{I_\epsilon} \\ 0 \end{bmatrix} U_{\text{sum}}$$

The Matlab code used for tuning the LQR for the elevation system can be found in chapter two of the appendix:

The weightings for the elevation controller were chosen by trial and error :

$$Q = \begin{bmatrix} 150 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 300 \end{bmatrix}$$

$$R = 5$$

This yielded the feedback gain matrix Ke:

$$Ke = \begin{bmatrix} 8.90 & 5.53 \end{bmatrix}$$

With the integral gain of Kei:

$$Kei = 7.64$$

2.5.2 Tuning the LQR for the Travel

LQR with integral action is used.
Augmenting the integral of the error as a state in the travel system gives the following system:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_5 \\ \dot{x}_6 \\ e \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ \frac{-m_H g L_P}{I_p} & -\frac{\mu_p}{I_p} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \frac{M(0)}{I_\tau(0)} & 0 & 0 & -\frac{u_\tau}{I_\tau(0)} & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_5 \\ x_6 \\ w \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{L_H}{I_p} \\ 0 \\ 0 \\ 0 \end{bmatrix} U_{\text{diff}}$$

The Matlab code used for tuning the LQR for the travel system can be found in Chapter 2 of the appendix.
The Q and R were chosen as follows:

$$Q = \begin{bmatrix} 50 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 0 & 200 \end{bmatrix}$$

$$R = 20$$

Yielding the feedback gain :

$$Kt = \begin{bmatrix} 2.1525 & 0.9643 & 7.1555 & 5.5498 \end{bmatrix}$$

With the integral gain of Kei:

$$Kti = 3.07$$

2.5.3 Tuning the LQR for the Pitch

An LQR controller is tuned for the pitch axis that will be deployed when using the joystick For the pitch, a forward gain is used for the LQR instead of the integral action, as it performed better in the operation of the joystick .
The Following is the state-space model of the pitch:

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{-m_H g L_P}{I_p} & -\frac{\mu_P}{I_p} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{L_M}{I_\epsilon} \end{bmatrix} U_{\text{diff}}$$

The Matlab code used for tuning the LQR for the pitch system can be found in chapter two of the appendix. The Q and R were chosen as follows :

$$Q = \begin{bmatrix} 50 & 0 \\ 0 & 30 \end{bmatrix}$$

$$R = 20$$

This yielded the optimal gain Kp:

$$Kp = \begin{bmatrix} 0.2478 & 1.135 \end{bmatrix}$$

With the forward gain of Kpf:

$$Kpf = 3.7$$

2.6 Conclusion

In conclusion, this chapter focused on the derivation of linearized models for designing a controller and emphasized the use of the LQR (Linear Quadratic Regulator) controller. The linearized models provide a simplified representation of the system dynamics, allowing for more straightforward controller design and analysis. By employing the LQR controller, an optimal control strategy can be achieved by minimizing a cost function that incorporates both state and control effort. The LQR controller design offers robustness and stability properties, making it a popular choice in various control applications. Overall, this chapter lays the foundation for effective controller design using linearized models and highlights the advantages of employing the LQR controller.

Chapter 3. Implementation

In this chapter, the realization of the control algorithms as well as the design of a user interface for the operation are considered. The introduction of the hardware used and the motivation for the different choices are presented.

3.1 LABView and the NI acquisition board

This section provides an in-depth examination of the utilization of LabVIEW software in the implementation of the controller, shedding light on its various roles in facilitating user interface and logging mechanisms for the operation of the controlled plant. LabVIEW, which stands for Laboratory Virtual Instrument Engineering Workbench, is a programming environment in which one can create programs using a graphical notation by connecting functional nodes via wires through which data flows[15]. In this regard, it differs from traditional programming languages like C, C++, or Java, in which you program with text. However, LabVIEW is much more than a programming language. It is an interactive program development and execution system designed for providing an easy development environment where the focus is on the core of the matter rather than dealing with unrelated technical difficulties.

3.2 The STM32F4 microcontroller

In this section, the topic is dedicated to the microcontroller used, namely stm32f412. This is a microcontroller from the manufacturer “STMicroelectronics” which is a global semiconductor company. The STM32F412 MCUs deliver the performance of Cortex®-M4 core with a floating point unit, running at 100 MHz, while achieving outstandingly low power consumption values in Run and Stop modes. It incorporates 14 timers, 16- and 32-bit, running at up to 100 MHz as well as USARTs running at up to 12.5 Mbit/s along with other communication protocols [16]. The mentioned set of characteristics are what is needed to implement the functionalities of acquisition and control. All of the development was conducted in STM32CUBEIDE which is an integrated development environment that simplifies the clock and different modules configurations. In the following sections, the configuration of the modules that were used will be discussed.

3.2.1 Hardware counters

One of the critical modules in our application is the counter as it allows to count the pulses of the three encoders as well as generate the required PWM signals for the electronic speed controllers. As mentioned previously the STM32 has 14 timers which have different characteristics and options. These counters are referred to as timers (TIM1 to TIM14) in the specification sheet. They are operated as timers when they are controlled by an external or an internal clock. Every timer has mainly three registers that are responsible for its operation. Namely, the PSC, the ARR and the CCR. Figure 3.1 show a diagram for a general-purpose timer.

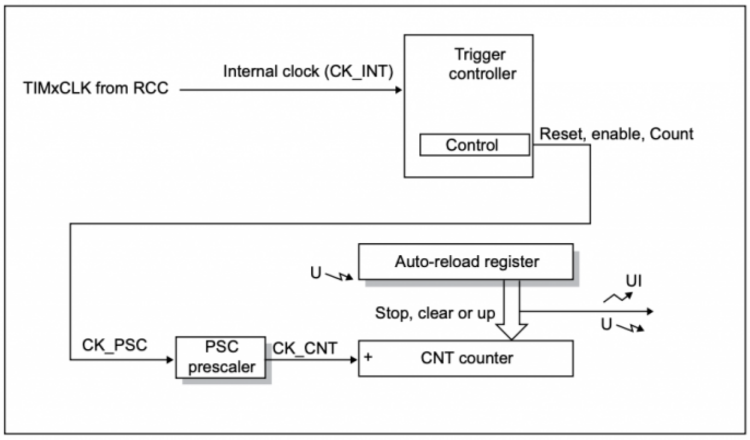


Figure 3.1. Simplified diagram showing the operation of a general purpose timer

The frequencies of the timers need to be adjusted to fit the need of the application.

$$F_{tim} = \frac{F_{clk}}{PSC * ARR}$$

Five timers are employed to cover all the functionalities where the 72Mhz clock is used for counting. The following table provides the setup and the operation for the timers:

Timer	Operation	PSC	ARR	CCR
TIM1	20ms for generating the PWM signal	400	3600	180 to 360
TIM2-TIM4	used for counting the increments of encoders	X	2^{16}	X
TIM14	10ms for synchronizing the LabView control loop	20	36000	X

Table 5. STM32F4 timers setup

3.2.2 Acquiring the three angles

One of the main drawbacks of the NI DAQ, when used in this project, is its lack of hardware counters. In total, four encoders are needed while the DAQ has only two. An attempt two program a software encoder using the digital pins of the NI DAQ board was done. It showed acceptable results when only the acquisition was running. However, when adding other calculations to the loop the NI DAQ started to miss some pulses of the encoder and consequently gives wrong readings which will propagate over time. As a result, a software encoder cannot be reliable in this application. A different approach for acquiring the encoder's data in real-time was successfully employed. The STM32F4 microcontroller was used for several purposes among which the counting of the encoder pulses using its hardware timers. Since at this stage the calculations are still mainly carried out in LABVIEW, this approach raises another problem of communication between the STM32 and LabVIEW. This challenge is tackled in section 3.2.5.

3.2.3 Generating the PWM signals

The NI DAQ originally lacked two counters: one for the third axis and another for generating the PWM signal. Consequently, the previous development team had to rely on an additional ADC to convert an analog signal to PWM, introducing redundancy. However, this redundancy was eliminated by leveraging the availability of counters in the STM32F4 microcontroller. The required PWM signals for the ESCs were generated using a hardware counter, thereby eliminating the need for the intermediary ADC conversion.

3.2.4 Reading the maneuvering actions

The joystick is incorporated to facilitate user interaction with the process, allowing for intuitive control and input adjustments. It serves as an input device enabling users to manipulate and interact with the system parameters effectively. The value from the joystick is acquired by the ADC (Analog-to-Digital Converter) in the STM32F4 microcontroller. Subsequently, the obtained value is transmitted via serial communication to a computer running LabVIEW software, where it is interpreted and utilized as set points for further processing and control.

3.2.5 Serial communication for the data and control action

Ensuring the precise and prompt arrival of data, encompassing both acquisition and control components, is of paramount importance. It is critical that the data reaches its destination accurately and without surpassing the maximum permissible delay, as any deviation beyond the established threshold can have significant degradation on the overall system performance and functionality. The digital sensors were used within the recommended standards, such as correct wire length and electrical characteristics, eliminating the need for incorporating filters. In order to avoid any data loss during the acquisition and the transfer of the two control actions communication between the computer and the MCU was set at a rate of 230400 bps. This is far less than the maximum bit rate that could be achieved in the STM32 which is 3.12 MBits/s [16]yet sufficient to transmit the data in the 3DOF in a relaxed manner. On the LabView side, there are several methods for transferring data with the MCU. However, employing conventional connections does not ensure real-time operation, as they are not designed specifically for this purpose. In contrast, LabVIEW VISA provides a solution that enables meeting the set baud rate and directly transferring data to a local variable, facilitating real-time data transmission through the usage of lower-level drivers [17]. This is crucial because LabVIEW will not only be utilized for logging purposes but also for control.

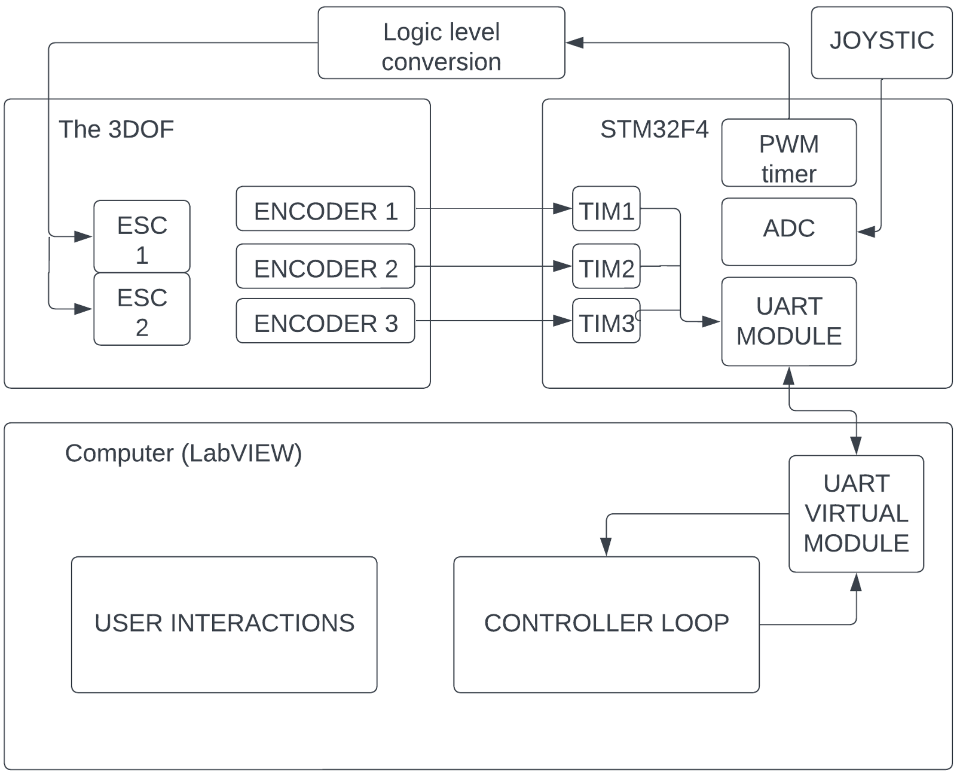
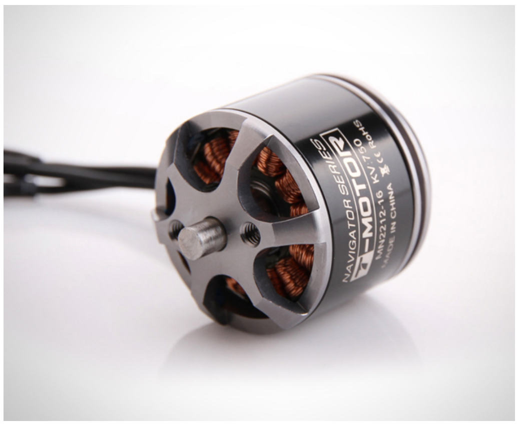


Figure 3.2. General architecture of the hardware configuration.

3.3 Thrust unit

Some implementations of this 3DOF bench helicopter incorporate PMDC motors[18] in which the results were acceptable. However, Due to the slow dynamics of these motors, they lead to a slower response. This is mitigated by using BLDC motors, which are known for their enhanced efficiency. This is proved by their use in small-scale aerial vehicles such as drones and remote control crafts. This is due to their high power-to-weight ratio, efficiency, durability, precise control, and compact size. These motors provide a significant amount of power relative to their weight, enabling aerial vehicles to achieve better performance and maneuverability. The brushless DC motor used in our small-scale helicopter is shown in figure 3.3a, it has a KV rating of 750 and operates at a voltage of 11.1 V, corresponding to a 3-cell LiPo battery commonly used in this type of applications. The selection of this voltage was made with future compatibility in mind, as it allows for easy integration with commercial batteries if a power supply is not available. By choosing this voltage, the control parameters can remain the same, ensuring ease of transition between power sources and providing flexibility for potential battery replacements. Another important module for the thrust unit is the Electronic Speed Controller or the ESC. The ESC module shown in figure 3.3b is a control system on its own and so it will control the motors to operate at the same speed under variable loads. Along with the speed controller the ESC incorporate a driver circuitry that comprises MOSFETs to provide the appropriate commutation for the stator coils. The commutation is based on the position of the rotor which could be achieved by processing the back electromagnetic field induced on the stator windings.



(a) 750KV brushless DC motor

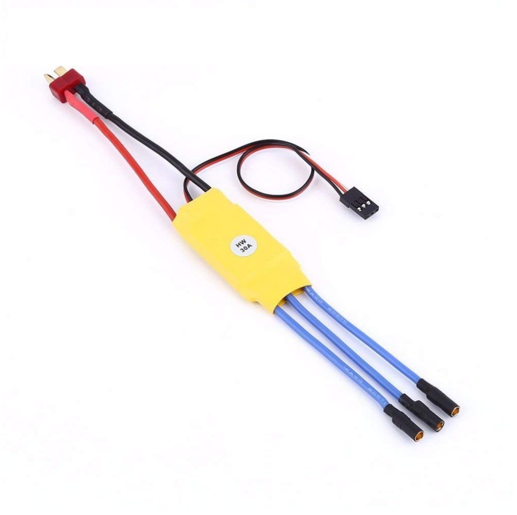
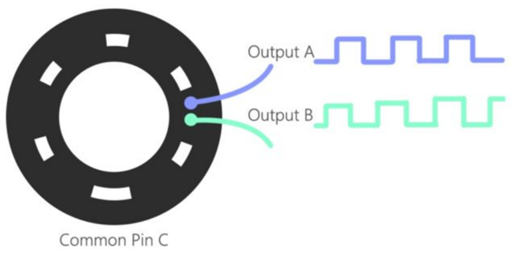

(b) 30A electronic speed controller

Figure 3.3. The thrust unit devices

3.4 The quadrature encoder

A rotary encoder, also called a shaft encoder, is an electro-mechanical device that converts the angular position or motion of a shaft or axle to digital output signals by using two light sources along with a disk of co-axially distances slits rotating with the shaft. When in motion the optical beam is either passed and detected by the phototransistor on the other side of the disk or blocked and hence undetected. This continuous passing and blocking of light generate two sets of pulses that are 90 degrees out of phase. The sensor used in the 3DOF bench is OMRON E6B2-CWZ6C shown in figure 3.4b, it is 1000 P/R [19]. Since only the increment of the angle can be known, an up-down counter must be used in order to track the pulses starting from a known physical position.



(a) Diagram showing the configuration of a generic incremental rotary encoder

(b) OMRON E6B2-CWZ6C 1000P/R rotary encoder

Figure 3.4. Sensor used in the 3DOF for the angle measurement

3.5 The state diagram for maneuvering using the joystick

Succeeding the development of the individual core controllers of the 3DOF, this section presents the flow and the logic of the operation. Until this point, the reference signals were provided in the software and this is the natural step for debugging and assessing the performance of the controller. It is important to reemphasize that one of the aims of this project is to imitate some of the situations which the CHINOOK helicopter encounters when transporting loads. In such a scenario, the helicopter has to maintain a particular height and pitching which corresponds to a specific elevation angle in the bench helicopter. Along that, it should maintain its desired pitch and travel. To achieve this, all of the controllers must be activated simultaneously. However, this cannot be realized between the pitch and the travel axes due to the under-actuation of the system where the difference in the thrust between the two propellers is the only input to both of these axes. For instance, given that the travel and the pitch are regulated, if the travel deviates from the desired position, the travel controller applies its calculated control action which will cause the pitch to deviate from its zero pitch and hence causes a conflict. The opposite scenario would raise the same issue, therefore, A middle solution must be implemented. In case the joystick is not moved the travel controller is active. when the pilot travel by moving the joystick side wise the pitch controller takes place, the state diagram for this operation is shown in figure 3.5 with its LabVIEW code on the next page.

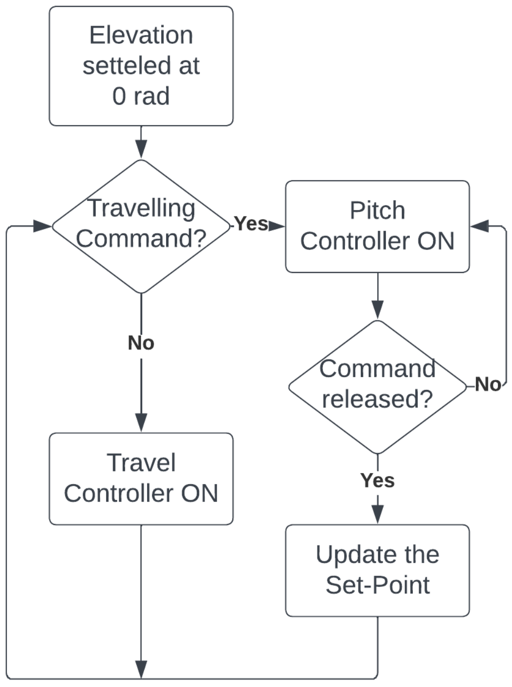


Figure 3.5. FlowChart for activating the pitch and the travel controllers when using the joystick

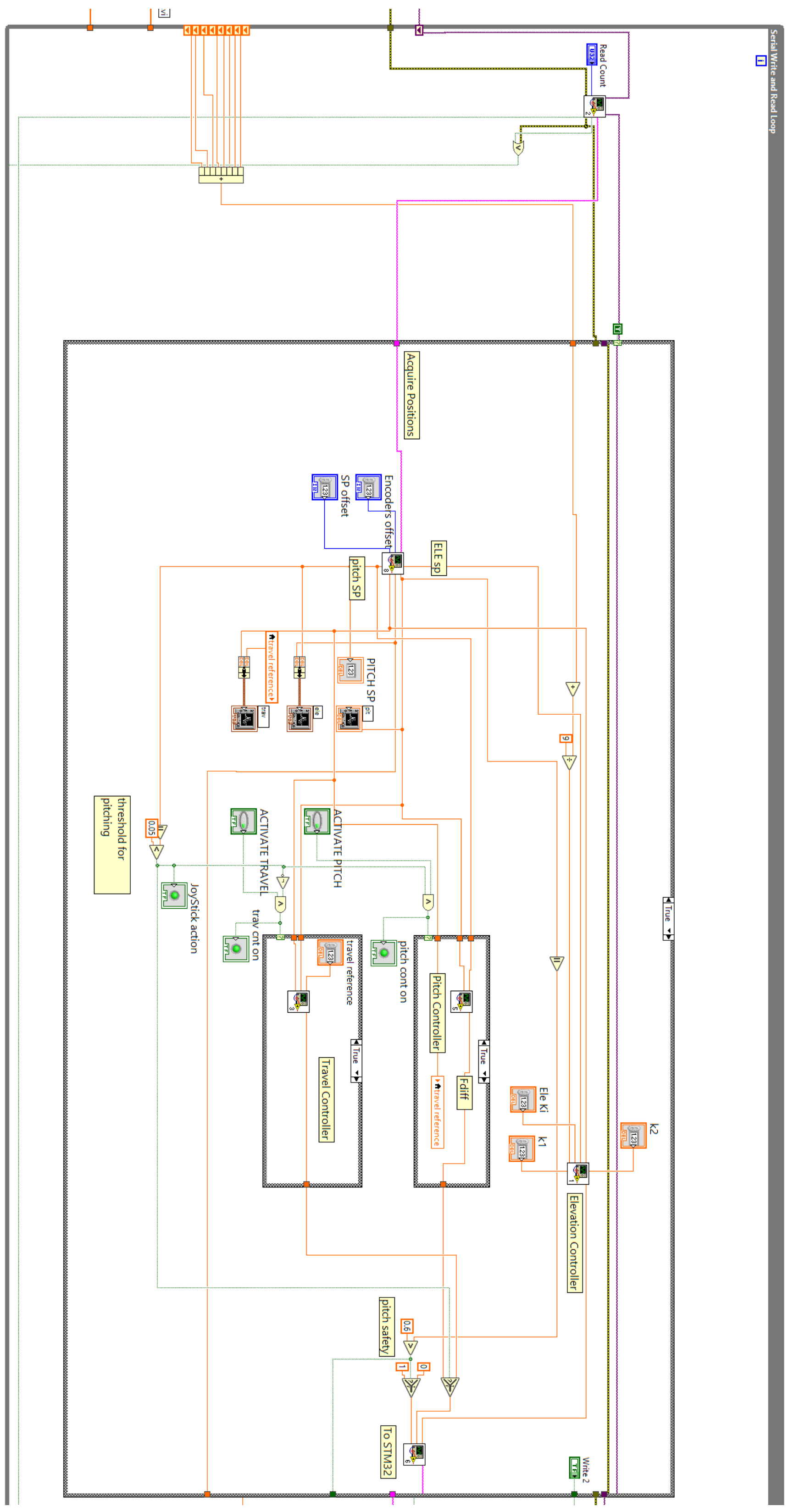


Figure 3.6. LabVIEW code for activating pitch and the travel controllers when using the joystick

3.6 Determination of the sampling time

The determination of the sampling time is based upon several factor. Ideally it is set to be as small as possible in order to get the same result as in the continuous model. However, the main factor from which the sampling frequency was set is the frequency of the ESCs. According to the manuals of the ESCs, the minimum frequency for updating the duty cycle of the ESCs and hence the speed of the propellers is 50 Hz. So calculating the control action more frequently would not introduce further improvements. The only factor that can render the controller more accurate is to synchronize the sampling interrupts with the interrupt of the PWM generator. The reason is that the thrust cannot be modified when the PWM counter is initiated. The CCR register is only updated at the end event of counting. The solution suggested is to initiate the control timer and then wait for a certain delay to trigger the PWM generation. The limitation is to keep enough time for the control action to be calculated before the new period starts. Furthermore, It has been verified that even when using an asynchronous 10ms control would result in a working controller for the elevation.

3.7 Implementing the controllers in labview

After all the necessary acquisitions have been set up, the attention is shifted towards the implementation of the controller within the control loop in LabVIEW. The LabVIEW loop is set such that it is triggered at every UART message arrival which is sent by the microcontroller each 10ms. Hence the synchronization is always met between the acquisition and the control algorithm and the application of the control action.
The elevation Controller The LabVIEW block corresponding to the LQR controller designed in Chapter 2 is shown below:

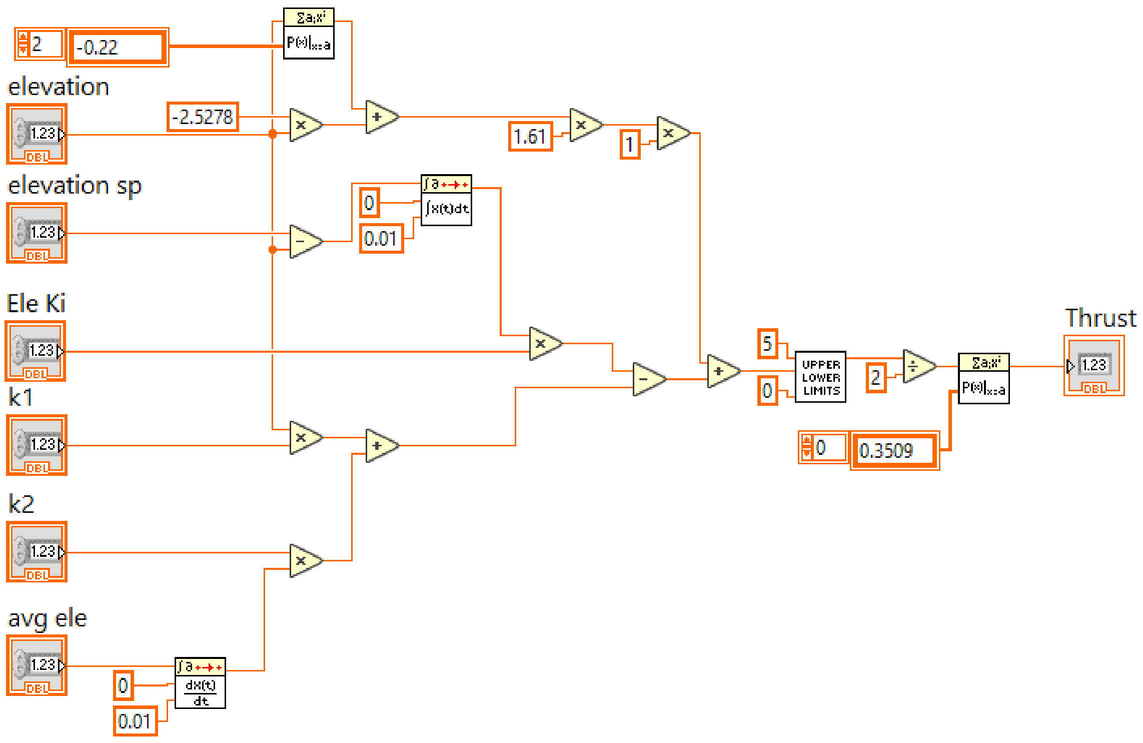


Figure 3.7. LabVIEW diagram for the elevation controller

The travel Controller: In order to ensure that the thrust difference does not lead the one motor to go beyond the allowed thrust or a negative thrust which is not possible, the thrust difference is limited to 2N.

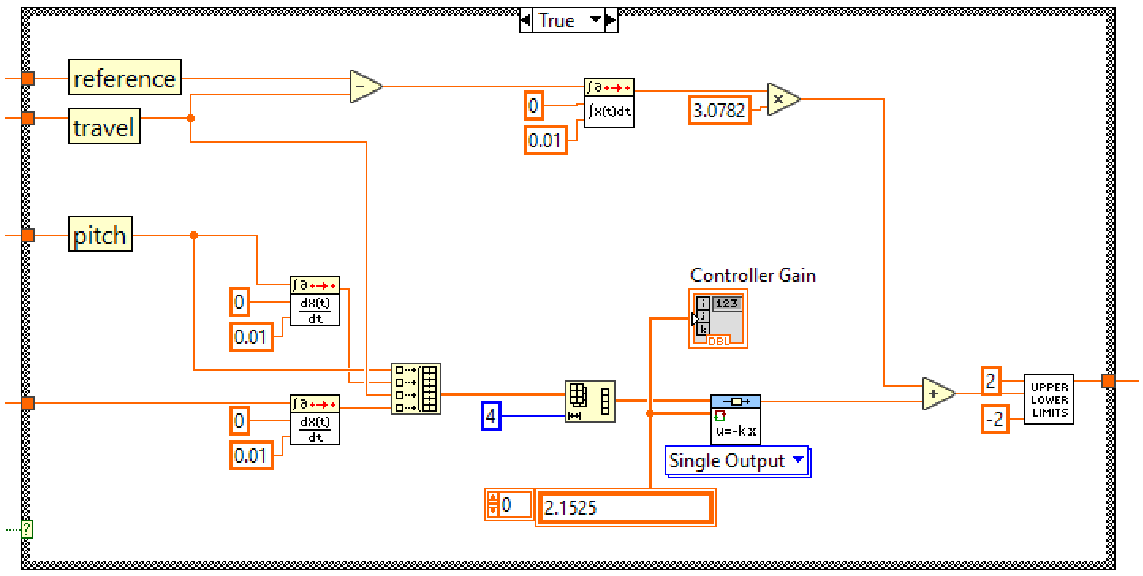


Figure 3.8. LabVIEW diagram for the travel controller

The pitch Controller:

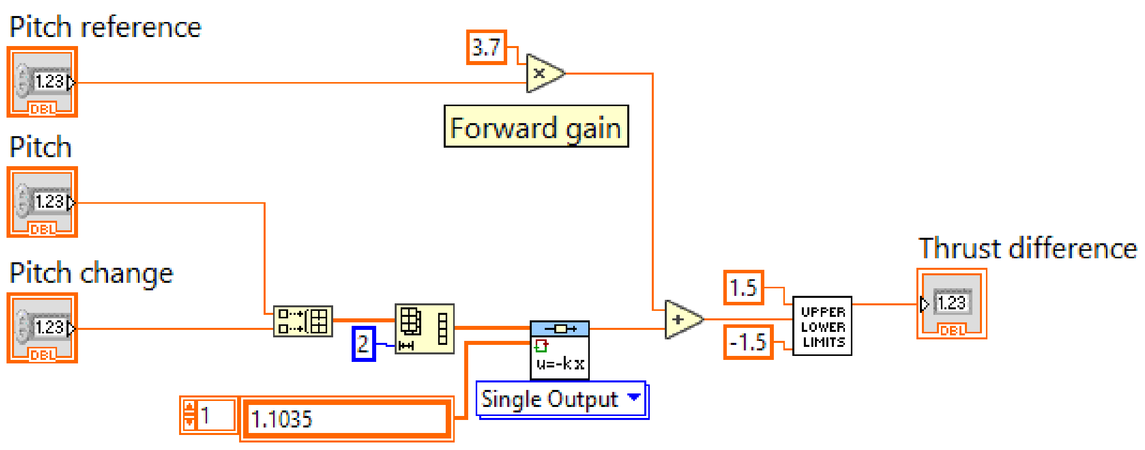
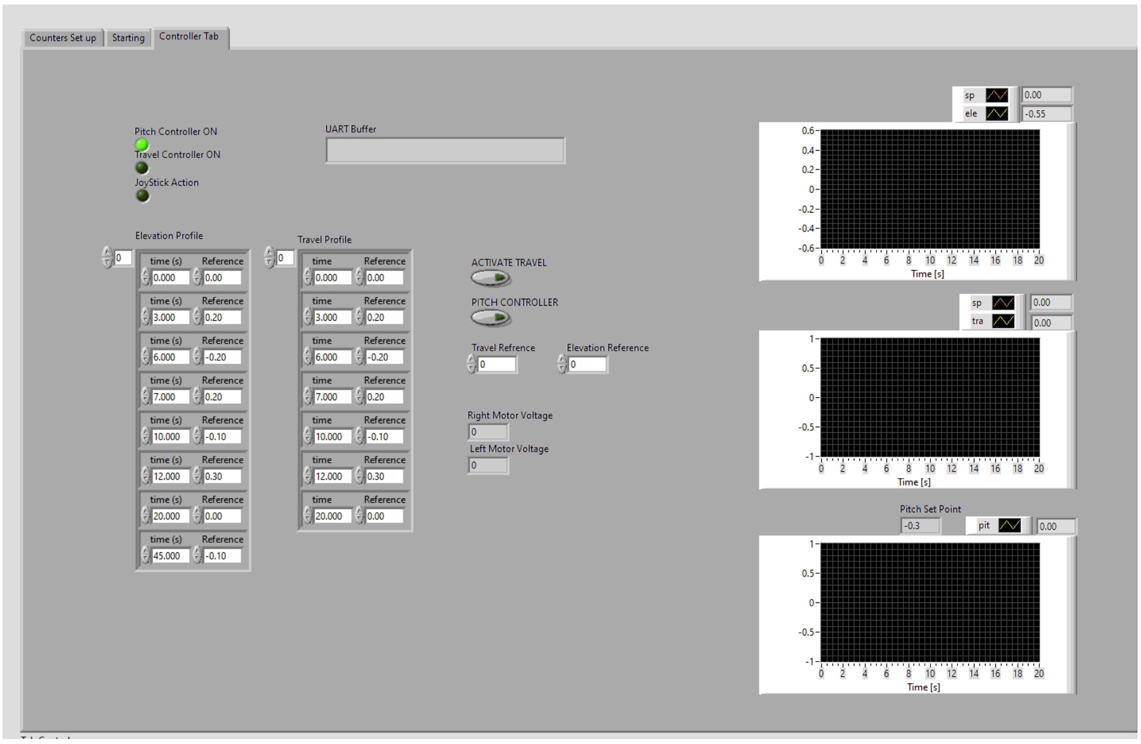


Figure 3.9. LabVIEW diagram for the pitch controller

3.8 The user interface

Having built all of the necessary controllers and hardware interaction software a user interface was designed such that only the critical values that directly affect the controller behavior are displayed for modification. This user interface aims to simplify the configuration of the communication and the performance of the initialization routine such as the initial hovering. It is designed to be self-explanatory, where it is divided into sections (tabs) corresponding to the largest setup step. In The first tab, the user is able to set up the communication with the acquisition board or with the microcontroller by choosing the port and the baud rate as well as the decoding of the message received. The second tab is named "Starting" in which the gains are set and the type of the set points inputs is chosen between a predefined profile or taken from the joystick. Once everything is set up the user clicks a launch the controller and automatically displays tab "Controller" tab. The chart for the three angles shows the set points and the real-time angles which allows us to get a visual sense of the performance of the controller and most importantly log the data for future analysis. Finally, the user can safely shut down the flight in case of undesired performance when testing different gains.



UART Buffer

Elevation Profile: 0

time (s)	Reference
0.000	0.00
3.000	0.20
6.000	-0.20
7.000	0.20
10.000	-0.10
12.000	0.30
20.000	0.00
45.000	-0.10

Travel Profile: 0

time	Reference
0.000	0.00
3.000	0.20
6.000	-0.20
7.000	0.20
10.000	-0.10
12.000	0.30
20.000	0.00

Travel Refrence: 0

Elevation Reference: 0

Right Motor Voltage: 0

Left Motor Voltage: 0

sp 0.00
ele -0.55

sp 0.00
tra 0.00

Pitch Set Point: -0.3
pit 0.00

Figure 3.10. Graphical user interface for the control phase

3.9 Conclusion

In conclusion, the implementation chapter has introduced the crucial components of both software and hardware necessary for the project. The utilization of the STM32F4 microcontroller, along with its various modules, such as timers, UART, and ADC, has enabled the achievement of the desired objectives. These modules played a vital role in designing and developing the controllers discussed in Chapter 2.

The combined efforts in implementing the hardware, and software gave the ability to do an analysis of the overall performance which will be held in the next chapter.

Chapter 4. Evaluation and discussion

The preceding chapter focused on three primary aspects: process modeling, controller design, and controller implementation using the STM32 microcontroller and NI LabVIEW. The objective of this chapter is to present the findings from these three components, including the measured data, as well as discuss the limitations encountered during each phase.

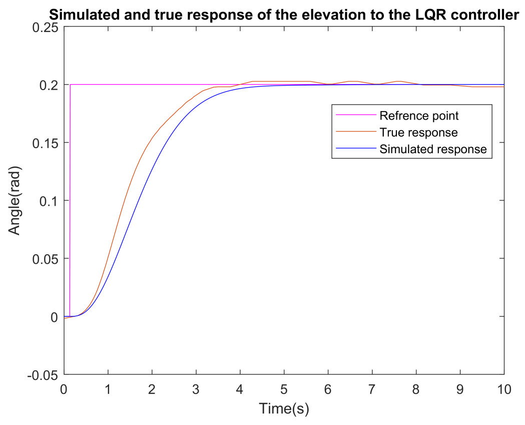
4.1 Usage of the UART for real-time control

In essence, the UART is designed for sending ASCII codes that are 8 bits wide, so transmitting a number would be received as its ASCII equivalent code. This redundant step implies using a decoder at the receiver end. Although that was the technique for sending the code from the microcontroller to the LabView VISA, the data which is 25 bytes was received with a negligible delay. Another communication of equal importance that has to take place is the transmission of the control actions to the microcontroller. In this case, the encoding /decoding issue is not encountered in acquiring the angles and the joystick maneuvers since the value for the control action can be directly sent as a byte rather than encoded in an ASCII format using Labview.

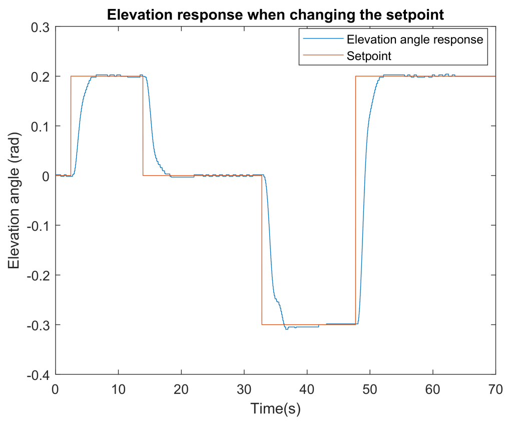
4.2 Controller Performance

In order to properly evaluate the performance of the overall system it is important to set criteria for doing so. This is chosen to be how well the controller performs when the system is operated with and without external disturbances. First, each axes was tested separately and then a disturbance that has a component along all of the axes was applied to see how the system return to the desired operating angle. First, the system's performance for the elevation axis was evaluated by setting the helicopter at 0 rad elevation angle, chosen as the setpoint. The recording in LabVIEW was initiated, capturing the system's response. Subsequently, the setpoint was changed to 0.2 rad, and the system's settling at the new setpoint was observed.

The recorded data from LabVIEW were then transferred to MATLAB and plotted on figure 4.1a and compared with the simulated data from the Simulink mode which can be found in appandix A.2. By plotting both sets of data on the same graph in figure 4.1. A visual comparison between the actual controlled plant and the simulated plant was made. This comparison allowed for an initial assessment of the system's performance.



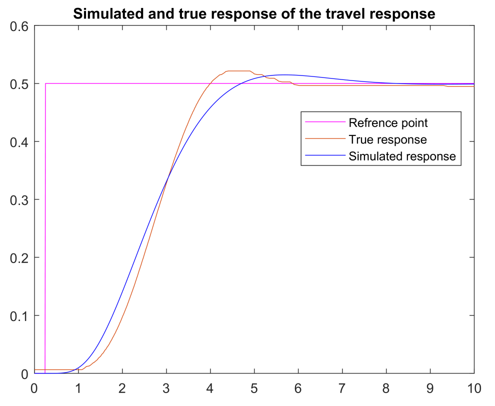
(a) Recorded versus simulated



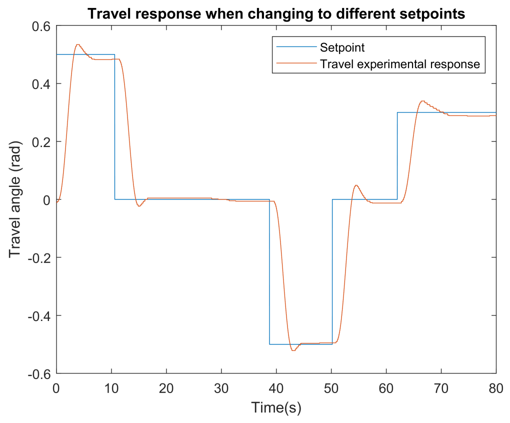
(b) Different setpoints on the elevation axis

Figure 4.1. Elevation axis responses

A similar procedure was conducted for the travel axis, with the system subjected to a step change in angle from 0 rad to 0.5 rad. The resulting data for the actual controlled plant and the simulated plant were plotted and compared in figure 4.2a. The error between the two datasets was calculated, specifically as the mean square error (MSE), providing a quantitative measure of the similarity or dissimilarity between the actual and simulated responses.



(a) Set point step change on the travel axis.



(b) Different setpoints on the travel axis.

Figure 4.2. Travel axis response

The same procedure was done for the Pitch, which resulted in the following graph:

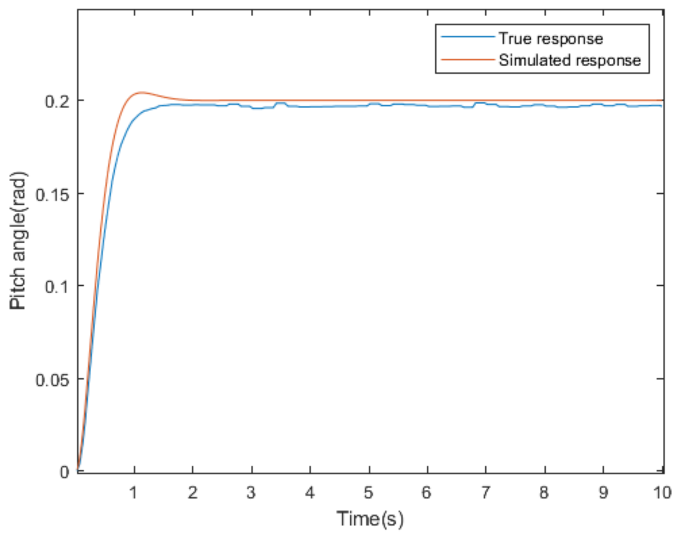


Figure 4.3. Response to a setpoint change in the pitch axis

To further evaluate the system's performance, several performance parameters were calculated. These parameters included overshoot, rise time, and mean square error (MSE), among others. The performance parameters were determined based on established criteria for evaluating control system performance.

Elevation

- Rise time: 2.5s
- Settling time: 4s (2% of steady state criterion)
- Overshoot: 0%
- MSE: 0.021rad

Travel

- Rise time 2.5s
- Settling time 5.5s (2% of the steady state criteria)
- Overshoot 4%
- MSE: 0.031rad

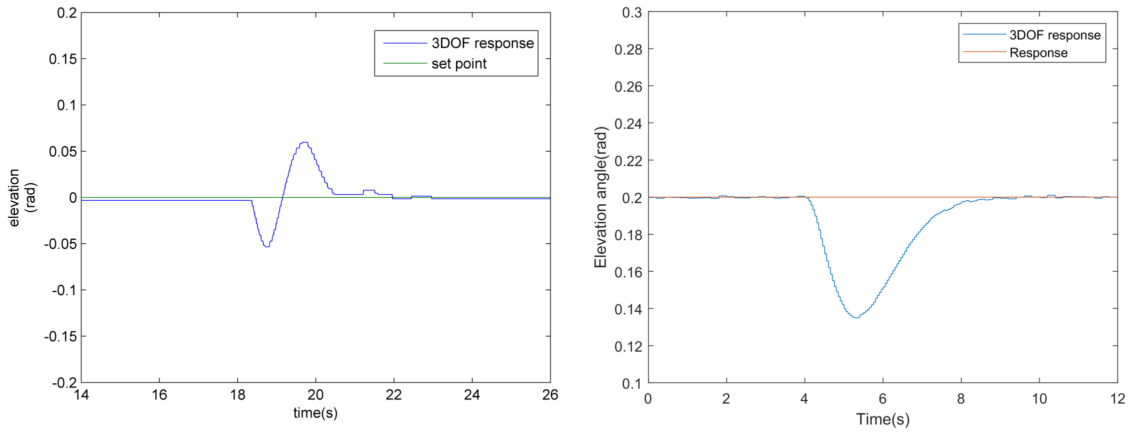
Pitch

- Rise time 1s
- Settling time 1.5s (2% of the steady state criteria)

- Overshoot 2.5%
- MSE: 0.015rad

Disturbed response To test how the system responds to unknown disturbances, a momentary downward disturbance was applied by hand to the elevation axes. the recorded response exported from LabVIEW to MATLAB is shown in figure 4.4a. The procedure was to wait for the helicopter to settle at its zero angle before starting to logging of the data and then the disturbance was applied at the 18^{th} second. After the initial angle is achieved the data was exported.

A second test for the disturbance was conducted by applying a momentary weight of 1N on the elevation rod. To conduct this test. The 3DOF was allowed to settle without any disturbance after that the recording started and a weight was placed on the elevation road until the elevation retracted its initial angle. The recorded result exported from LabVIEW are shown in figure 4.4b.



(a) momentary disturbance (b) permanent loading

Figure 4.4. Elevation Disturbance Response

To test the recovery of the desired travel when disturbed, the 3DOF is momentarily pushed in its positive direction. The results recorded are depicted in figure 4.5.

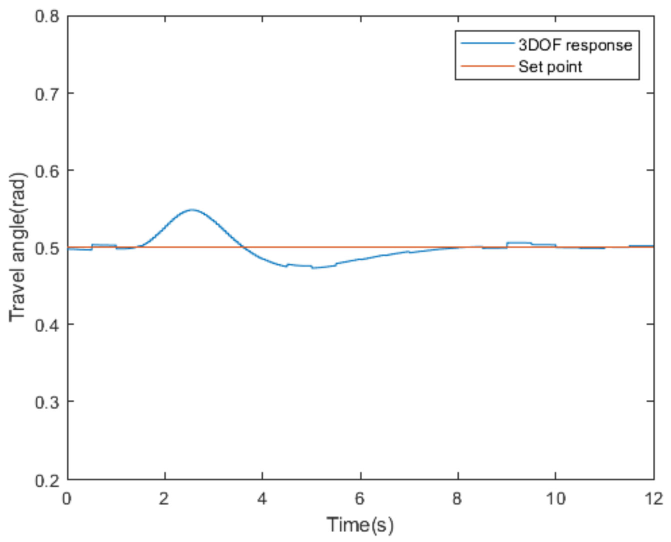


Figure 4.5. Travel disturbance response

The recovery time is defined as the time needed for the elevation and the travel to return to the desired set point

- Recovery time for the momentary disturbance on elevation: 2s.
- Recovery time for the 1N disturbance on elevation: 4s.
- Recovery time for the momentary disturbance on the travel: 6s.

The analysis of the performance parameters and the comparison of the actual and simulated responses revealed the validity of the model, as it consistently produced similar results to the actual controlled plant. This conclusion was supported by the close agreement between the two sets of data and the calculated performance parameters.

4.3 Improvements

This section serves as a summary of the notable improvements that have been accomplished as a result of the work conducted in this project.

1. The developed system now possesses the capability to read all three angles simultaneously, allowing for a more comprehensive analysis of spatial orientation.

2. Through model improvements, significant enhancements have been achieved in the system, leading to notable performance enhancements across all axes.

3. By employing the feedback linearization technique, the elevation axes of the system can

now be operated throughout their entire range, thereby expanding the system's operational capabilities.

4. The implementation of a realistic maneuvering mechanism using a joystick has been successfully incorporated into the system, enabling intuitive and precise control during operations.

5. The removal of hardware redundancies, including the ADC (Analog-to-Digital Converter), has resulted in notable improvements in both the overall performance and cost-effectiveness of the system.

Conclusion

In conclusion, this thesis has successfully achieved several milestones in the development of the 3DOF plateform.

Firstly, a workable model for the three axes of freedom was obtained using both linear and non-linear identification method. This laid the foundation for further advancements in the control and acquisition of the system's angles. Next, a feedback linearization technique was employed to effectively reduce the non-linearities present in the elevation axis. This allowed for the design of linear controllers for both subsystems, enhancing the overall control performance.

To address the challenge of acquiring the three angles, exploitation of the timers in the STM32F4 microcontroller was used to acheive real time acquisition. After that a real time communication between the STM32F4 was succesfully established. Using this microcontroller not only yielded the precise measurement of angles but also enabled the design of a communication technique between LabVIEW and the microcontroller.

Furthermore, by eliminating the redundancy of using the NI acquisition board and an ADC for generating PWM signals, a source of delay and synchronization issues was removed. This optimization not only improved the system's performance but also significantly reduced the overall cost of the setup. Finally, a testing and evaluation were conducted using the new setup. The system demonstrated favorable results, validating the effectiveness of the proposed techniques and highlighting the successful achievement of the project objectives.

Although the LQR controllers present in this thesis were execlusively implemented in LabView, a successful attempt to implement the controller directly was successful. The controller was implemented for the elevation controller, this shows the possibility for commissioning the flight controller directly on the 3DOF without the usage of a computer.

In summary, this project has contributed to the development of a robust and efficient system, showcasing advancements in modeling, control design, position acquisition, and overall system optimization. The outcomes of this research have paved the way for future improvements and applications in the field.

Future Work In this chapter, we outline several key ideas for future work and improvements in the field. These ideas aim to enhance the performance and capabilities of the existing system.

1. Implementation of a Standalone Controller: One area of focus for future work is the development of a standalone controller for the system. Currently, the system relies on a connected computer for control and operation. By designing and implementing a standalone controller, the system can be made more independent and portable. This will allow for greater flexibility in its applications and reduce the reliance on external devices.

2. Improved Joystick Performance: Another aspect that requires attention is the improvement of joystick performance. The joystick serves as the primary input device for controlling the system. Enhancing its sensitivity, precision, and response time will result in a more intuitive and accurate control experience. This can be achieved through hardware enhancements or advanced signal processing algorithms.

3. Integration of an Inertial Measurement Unit (IMU): To reduce the number of wire connections and enhance the system's mobility, the incorporation of an Inertial Measurement Unit (IMU) for orientation measurement is a promising avenue for future work. By utilizing an IMU, which consists of sensors like accelerometers and gyroscopes, the system can obtain orientation information without relying on wired encoders. This will improve the system's flexibility, reduce complexity, and potentially enhance its performance.

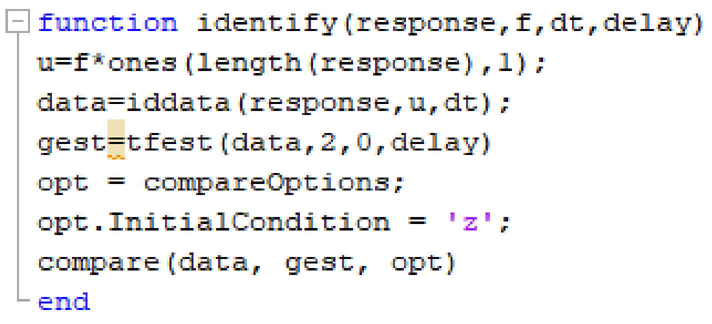
4. Embedded Controller with STM32: The current system, due to its wired encoder configuration, has limitations in performing full rotations. To address this issue and enable uninterrupted operation, integrating an embedded controller, such as STM32, can be explored. By incorporating an embedded controller along with an IMU and utilizing a 360-degree power connector, the system can achieve continuous rotation capability. This will not only enhance the system's functionality but also mitigate the risk of damaging the helicopter by reaching its maximum travel limit.

5. Development of Advanced Controllers: Furthermore, future work can involve the development of more advanced controllers, such as Model Predictive Control (MPC). MPC is a control technique that utilizes a model of the system and considers future predictions to generate optimal control inputs. Implementing an MPC controller can improve the system's stability, response time, and trajectory tracking accuracy. This opens up possibilities for more sophisticated control strategies and potential performance enhancements.

Chapter A. Appendix

A.1 Model identification

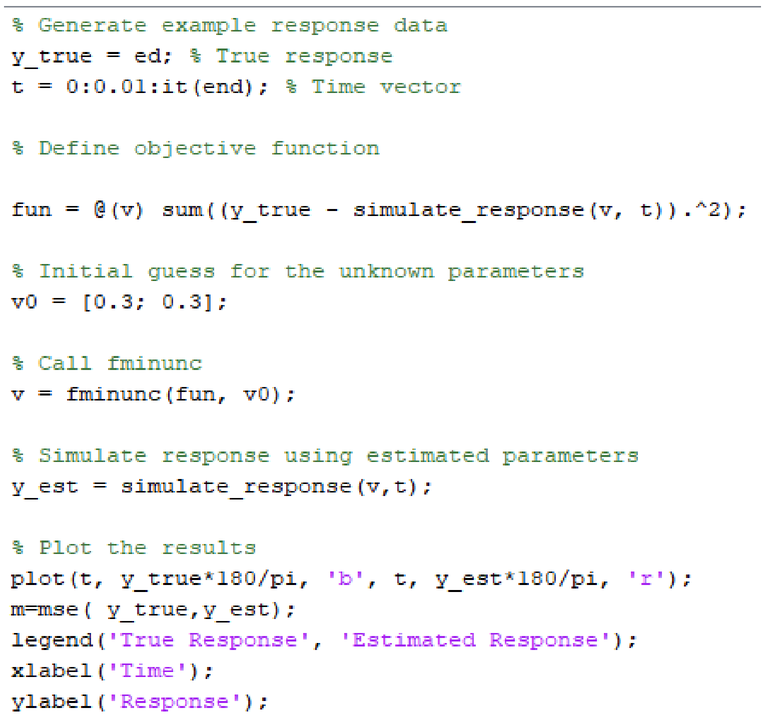
A.1.1 Code used in pitch identification



```
function identify(response,f,dt,delay)
u=f*ones(length(response),1);
data=iddata(response,u,dt);
gest=tfest(data,2,0,delay)
opt = compareOptions;
opt.InitialCondition = 'z';
compare(data, gest, opt)
end
```

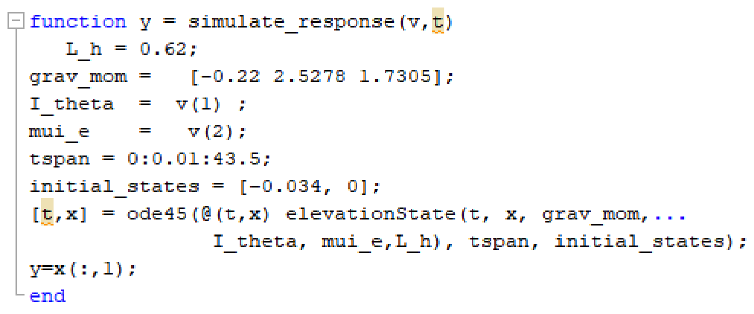
Figure A.1. System identification code

A.1.2 Codes used in elevation identification



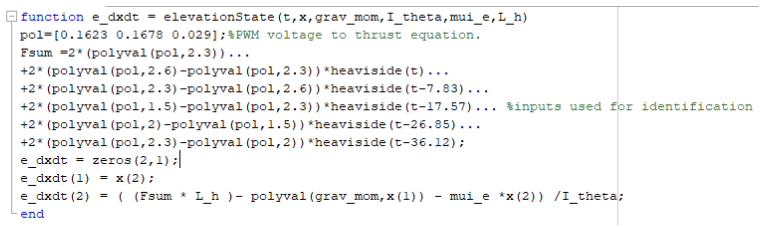
```
% Generate example response data
y_true = ed; % True response
t = 0:0.01:it(end); % Time vector

% Define objective function

fun = @(v) sum((y_true - simulate_response(v, t)).^2);

% Initial guess for the unknown parameters
v0 = [0.3; 0.3];

% Call fminunc
v = fminunc(fun, v0);

% Simulate response using estimated parameters
y_est = simulate_response(v,t);

% Plot the results
plot(t, y_true*180/pi, 'b', t, y_est*180/pi, 'r');
m=mse( y_true,y_est);
legend('True Response', 'Estimated Response');
xlabel('Time');
ylabel('Response');
```

Figure A.2. The Matlab code for the NLLS algorithm



```
function y = simulate_response(v,t)
   L_h = 0.62;
grav_mom =   [-0.22 2.5278 1.7305];
I_theta  =  v(1) ;
mui_e    =   v(2);
tspan = 0:0.01:43.5;
initial_states = [-0.034, 0];
[t,x] = ode45(@(t,x) elevationState(t, x, grav_mom,...
               I_theta, mui_e,L_h), tspan, initial_states);
y=x(:,1);
end
```

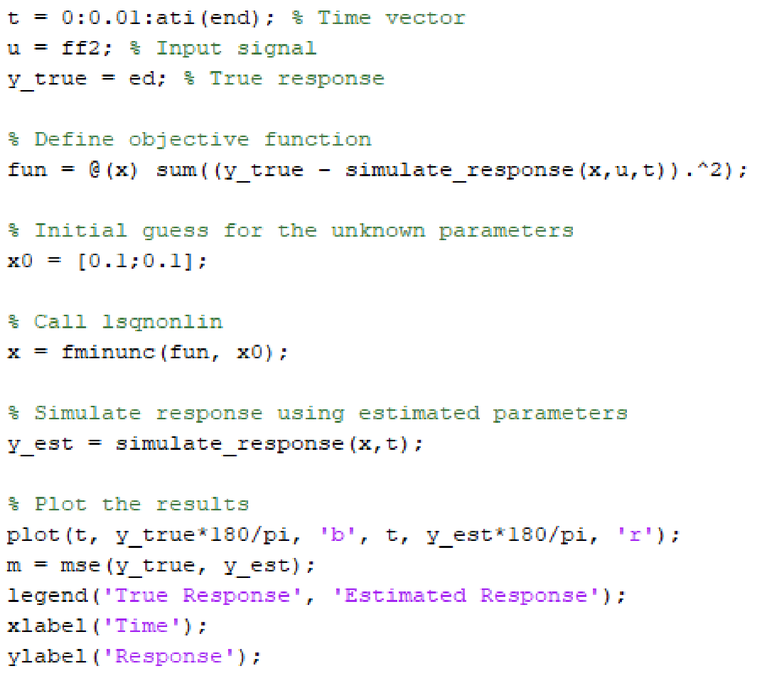
Figure A.3. The code of the included simulated response function.



```
function e_dxdt = elevationState(t,x,grav_mom,I_theta,mui_e,L_h)
pol=[0.1623 0.1678 0.029];%PWM voltage to thrust equation.
Fsum =2*(polyval(pol,2.3))...
+2*(polyval(pol,2.6)-polyval(pol,2.3))*heaviside(t)...
+2*(polyval(pol,2.3)-polyval(pol,2.6))*heaviside(t-7.83)...
+2*(polyval(pol,1.5)-polyval(pol,2.3))*heaviside(t-17.57)... %inputs used for identification
+2*(polyval(pol,2)-polyval(pol,1.5))*heaviside(t-26.85)...
+2*(polyval(pol,2.3)-polyval(pol,2))*heaviside(t-36.12);
e_dxdt = zeros(2,1);
e_dxdt(1) = x(2);
e_dxdt(2) = ( (Fsum * L_h )- polyval(grav_mom,x(1)) - mui_e *x(2)) /I_theta;
end
```

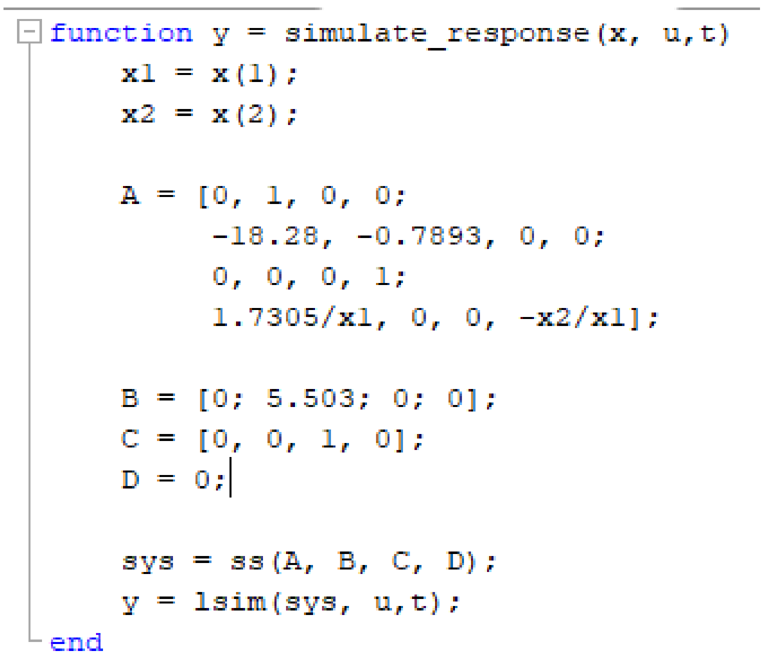
Figure A.4. The code of simulating the nonlinear elevation equation.

A.1.3 Codes used in travel identification



```
t = 0:0.01:ati(end); % Time vector
u = ff2; % Input signal
y_true = ed; % True response

% Define objective function
fun = @(x) sum((y_true - simulate_response(x,u,t)).^2);

% Initial guess for the unknown parameters
x0 = [0.1;0.1];

% Call lsqnonlin
x = fminunc(fun, x0);

% Simulate response using estimated parameters
y_est = simulate_response(x,t);

% Plot the results
plot(t, y_true*180/pi, 'b', t, y_est*180/pi, 'r');
m = mse(y_true, y_est);
legend('True Response', 'Estimated Response');
xlabel('Time');
ylabel('Response');
```

Figure A.5. The code of used for travel identification



```
function y = simulate_response(x, u,t)
    x1 = x(1);
    x2 = x(2);

    A = [0, 1, 0, 0;
         -18.28, -0.7893, 0, 0;
         0, 0, 0, 1;
         1.7305/x1, 0, 0, -x2/x1];

    B = [0; 5.503; 0; 0];
    C = [0, 0, 1, 0];
    D = 0;

    sys = ss(A, B, C, D);
    y = lsim(sys, u,t);
end
```

Figure A.6. The code used to estimate the travel parameters.

A.2 Control design

A.2.1 Simulink code for simulating the elevation nonlinear system

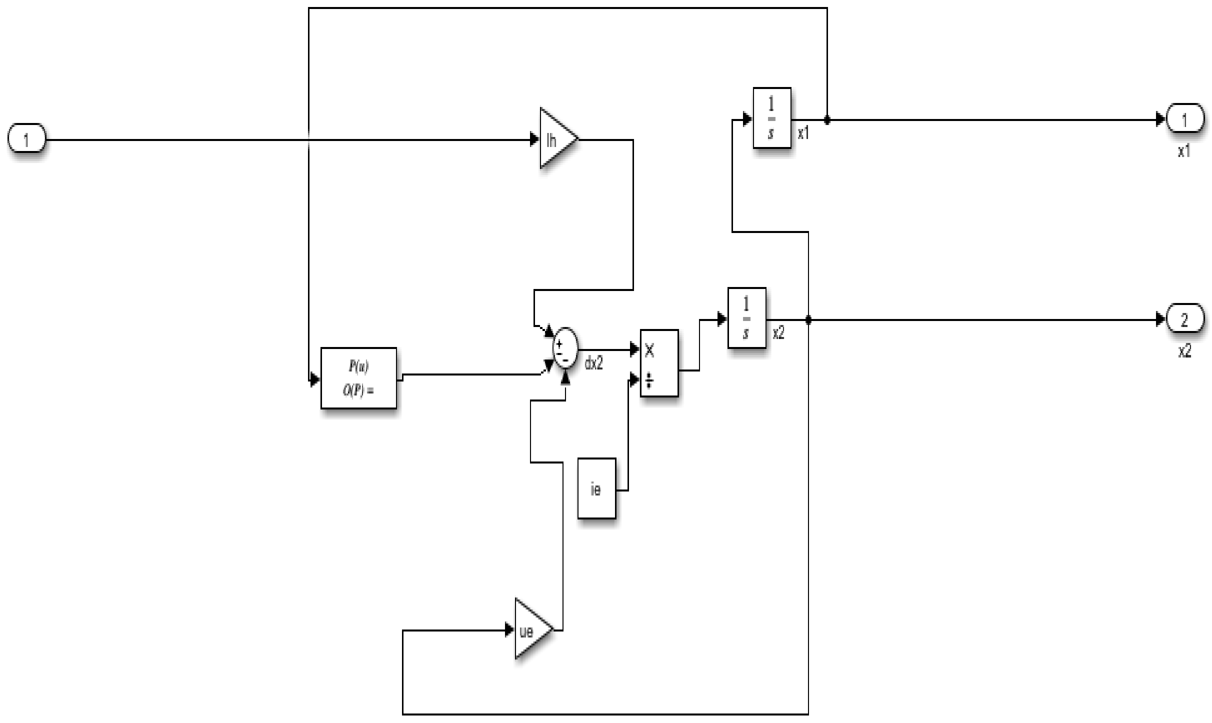


Figure A.7. Simulink code for simulating the elevation nonlinear system

A.2.2 Simulink code for feedback linearization block.

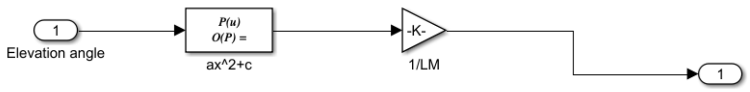


Figure A.8. Simulink code for feedback linearization block.

A.2.3 Simulink code for Comparing the linear and nonlinear elevation systems.

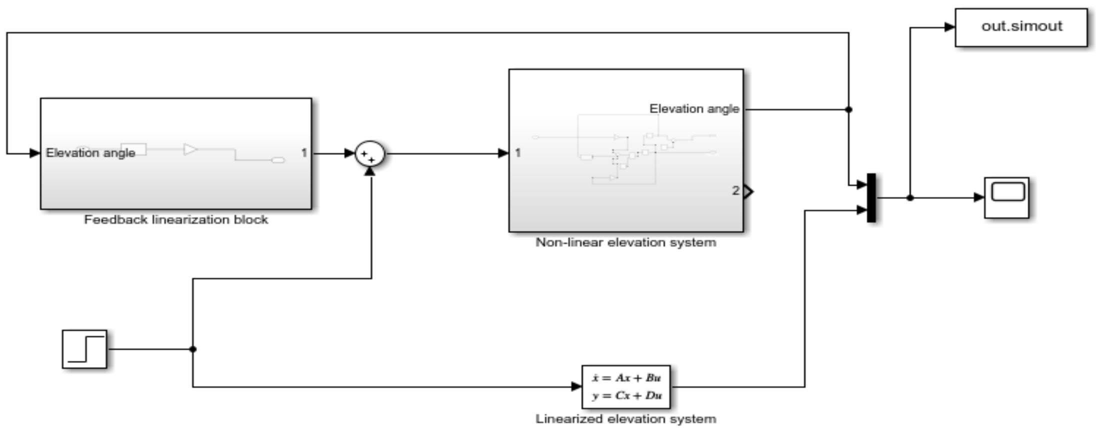
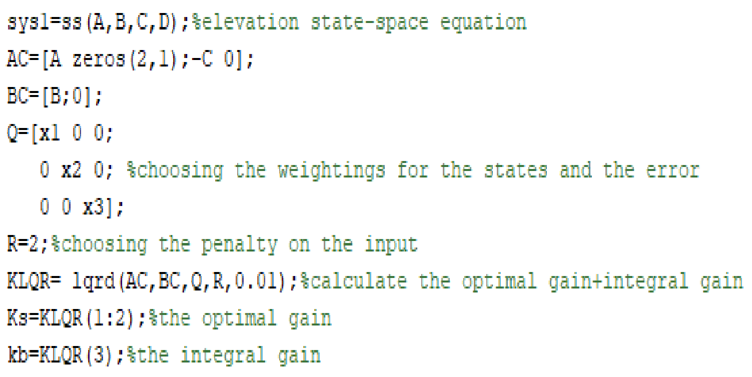

out.simout
Elevation angle
1
Feedback linearization block
Elevation angle
1
2
Non-linear elevation system
$\dot{x} = Ax + Bu$
$y = Cx + Du$
Linearized elevation system

Figure A.9. Simulink code for Comparing the linear and nonlinear elevation systems.

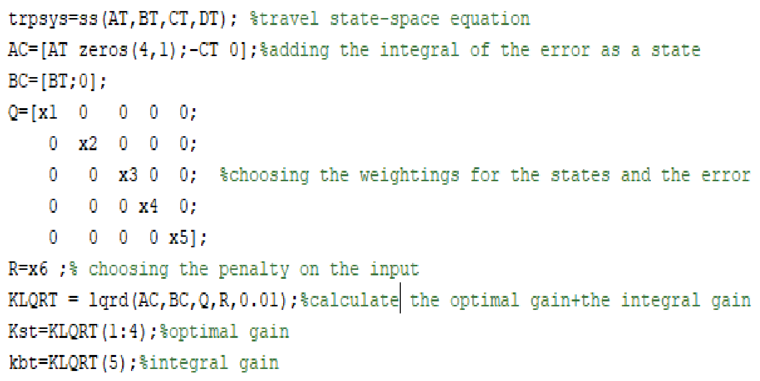
A.2.4 Matlab code for tuning the LQR for the elevation.



```
sys1=ss(A,B,C,D);%elevation state-space equation
AC=[A zeros(2,1);-C 0];
BC=[B;0];
Q=[x1 0 0;
   0 x2 0; %choosing the weightings for the states and the error
   0 0 x3];
R=2;%choosing the penalty on the input
KLQR= lqrd(AC,BC,Q,R,0.01);%calculate the optimal gain+integral gain
Ks=KLQR(1:2);%the optimal gain
kb=KLQR(3);%the integral gain
```

Figure A.10. Matlab code for the tuning of LQR for the elevation system.

A.2.5 Matlab code for tuning the LQR for the travel.

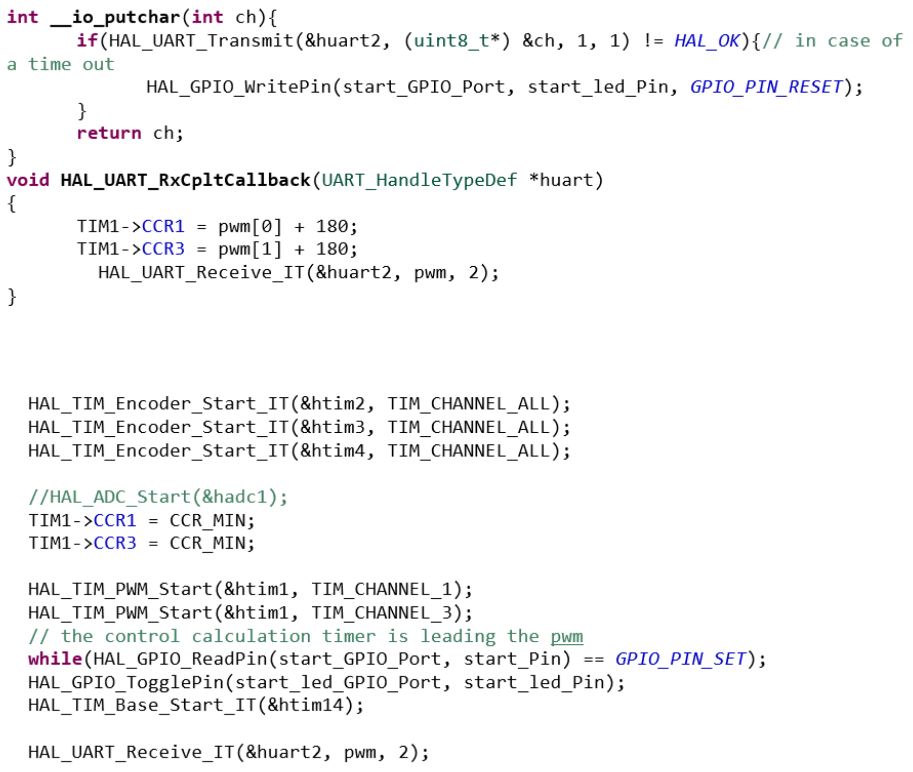


```
trpsys=ss(AT,BT,CT,DT); %travel state-space equation
AC=[AT zeros(4,1);-CT 0];%adding the integral of the error as a state
BC=[BT;0];
Q=[x1  0   0  0  0;
    0  x2  0  0  0;
    0   0  x3 0  0;  %choosing the weightings for the states and the error
    0   0  0 x4  0;
    0   0  0  0 x5];
R=x6 ;% choosing the penalty on the input
KLQRT = lqrd(AC,BC,Q,R,0.01);%calculate the optimal gain+the integral gain
Kst=KLQRT(1:4);%optimal gain
kbt=KLQRT(5);%integral gain
```

Figure A.11. Matlab code for the tuning of LQR for the travel system.

A.3 Implementation:

A.3.1 STM code used for data acquisition, PWM generation and communication



```
int __io_putchar(int ch){
        if(HAL_UART_Transmit(&huart2, (uint8_t*) &ch, 1, 1) != HAL_OK){// in case of
a time out
                HAL_GPIO_WritePin(start_GPIO_Port, start_led_Pin, GPIO_PIN_RESET);
        }
        return ch;
}
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
        TIM1->CCR1 = pwm[0] + 180;
        TIM1->CCR3 = pwm[1] + 180;
          HAL_UART_Receive_IT(&huart2, pwm, 2);
}



  HAL_TIM_Encoder_Start_IT(&htim2, TIM_CHANNEL_ALL);
  HAL_TIM_Encoder_Start_IT(&htim3, TIM_CHANNEL_ALL);
  HAL_TIM_Encoder_Start_IT(&htim4, TIM_CHANNEL_ALL);

  //HAL_ADC_Start(&hadc1);
  TIM1->CCR1 = CCR_MIN;
  TIM1->CCR3 = CCR_MIN;

  HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
  HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_3);
  // the control calculation timer is leading the pwm
  while(HAL_GPIO_ReadPin(start_GPIO_Port, start_Pin) == GPIO_PIN_SET);
  HAL_GPIO_TogglePin(start_led_GPIO_Port, start_led_Pin);
  HAL_TIM_Base_Start_IT(&htim14);

  HAL_UART_Receive_IT(&huart2, pwm, 2);
```

Figure A.12. STM code used for data acquisition, PWM generation and communication

A.3.2 Interrupt service routines for the acquisition and control synchronization

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){
      HAL_GPIO_TogglePin(timer_int_GPIO_Port, timer_int_Pin);
      printf("%+0*d%+0*d%+0*d%+0*d%+0*d\n",5,(int16_t)e,5,(int16_t)t,5,(int16_t)p,
5,(int16_t)ele_sp,5,(int16_t)pit_sp);
       HAL_ADCEx_InjectedStart(&hadc1);
      ele_sp = HAL_ADCEx_InjectedGetValue(&hadc1, ADC_INJECTED_RANK_1);
      pit_sp = HAL_ADCEx_InjectedGetValue(&hadc1, ADC_INJECTED_RANK_2);
}


void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim){
      if(htim == &htim2 )
            e = __HAL_TIM_GET_COUNTER(htim);
      if(htim == &htim3)
            p= __HAL_TIM_GET_COUNTER(htim);
      if(htim == &htim4)
            t = __HAL_TIM_GET_COUNTER(htim);
}
```

Figure A.13. Interrupt service routines for the aqcuisition and control synchronization

A.3.3 Simulink code for simulationg the LQR controller for the elevation system:

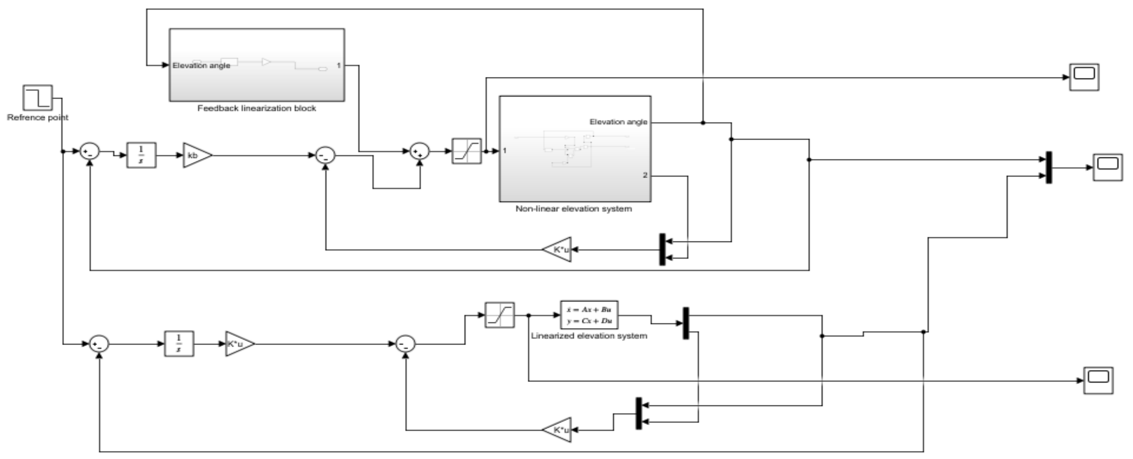


Figure A.14. Simulink code for simulation the LQR controller for the elevation system:

A.3.4 Simulink code for simulation of the LQR controller for the travel system:

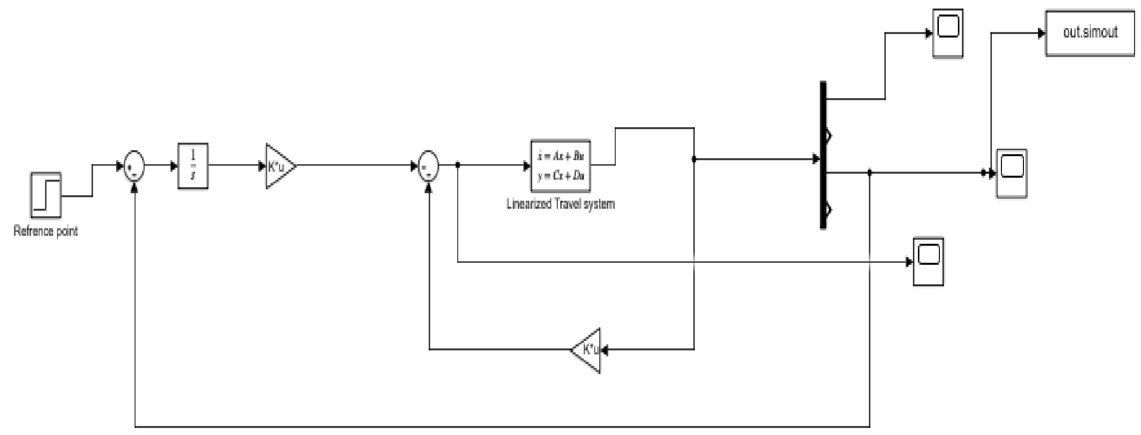


Figure A.15. Simulink code for simulating the LQR controller for the travel system:

References

[1] Quanser Inc. *3 DOF Helicopter*. en. Accessed on April 2023. 2015. URL: https://www.quanser.com/products/3-dof-helicopter/.

[2] National Instruments. *NI 6221 Specifications*. en. Accessed on April 2023. URL: https://www.ni.com/docs/en-US/bundle/pci-pxi-usb-6221-specs/page/specs.

[3] *Arduino UNO R3*. en. Accessed on May 2023. URL: https://docs.arduino.cc/hardware/uno-rev3.

[4] *MATLAB*. en. Accessed on May 2023. URL: https://www.mathworks.com/products/matlab.html.

[5] Erik Bodin and Fanny Stenholm. "Modeling & Control of a 3DOF Helicopter". en. Linköping: Linköping University, 2015.

[6] Mathworks. *tfest to Estimate transfer function model*. en. Accessed on May 2023. URL: https://www.mathworks.com/help/ident/ref/tfest.html.

[7] A.E. Pearson. "Least squares parameter identification of nonlinear differential I/O models". In: *Proceedings of the 27th IEEE Conference on Decision and Control*. Dec. 1988, 1831–1835 vol.3.

[8] Mathworks. *fminunc*. en. Accessed on april 2023. URL: https://www.mathworks.com/help/optim/ug/fminunc.html.

[9] Li Weiping Jean-Jacques E Slotine. *Applied Nonlinear Control*. en. Prentice Hall, 1991, pp. 208–213.

[10] *Control Tutorials for MATLAB and Simulink (CTMS)*. en. Accessed on may 2023. URL: https://ctms.engin.umich.edu/CTMS/index.php?.

[11] Prof. Alberto Bemporad. *Integral action in state feedback control*. University of trento, 2010-2011.

[12] M. Chidambaram Hanmant G. Malkapure. "Comparison of Two Methods of Incorporating an Integral Action in Linear Quadratic Regulator". In: Third International Conference on Advances in Control and Optimization of Dynamical Systems. Kanpur, India, 2014.

[13] Mathworks. *Design discrete linear-quadratic (LQ) regulator for continuous plant*. en. URL: https://www.mathworks.com/help/control/ref/lqrd.html.

[14] Kamran Iqbal. *Controllers for Discrete State Variable Models*. en. Accessed on may 2023. University of Arkansas at Little Rock. URL: https://eng.libretexts.org/Bookshelves/Industrial_and_Systems_Engineering/Book%3A_Introduction_to_Control_Systems_%28Iqbal%29/10%3A_Controllers_for_Discrete_State_Variable_Models/10.02%3A_Controllers_for_Discrete_State_Variable_Models.

[15] Jeffrey Travis and Jim Kring. *LabVIEW for everyone: Graphical programming made easy and fun*. en. Prentice-Hall PTR, 2007.

[16] *STM32F412xE STM32F412xG Datasheet*. Retrieved from https://www.st.com/ resource/en/datasheet/stm32f412cg.pdf. Accessed on May 17, 2023.

[17] *NI-VISA overview*. Retrieved from https://www.ni.com/en-lb/support/documentation/supplemental/06/ni-visa-overview.html#section-1826635511. Accessed on May 17, 2023.

[18] Made-for-science. *Made for science-Quanser 3DOFhelicopter - UserManual*. en.

[19] *Incremental Rotary Encoder OD 40 dia. E6B2-CWZ6C*. OMRON.