RAM ROBOTICS AND MECHATRONICS

MULTIROTORS EMBEDDING ELASTIC SENSORS TO MEASURE ARM DEFLECTION: SYSTEM MODELING AND CONTROL

S. (Salma) Bahaa El Din Hassan Abdelhalim

BSC ASSIGNMENT

Committee:
prof. dr. ir. A. Franchi
C. Gabellieri, Ph.D
dr. ir. D. Alveringh

July, 2022

031RaM2022
Robotics and Mechatronics
EEMCS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE. | TECHMED CENTRE

UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

MULTIROTORS EMBEDDING ELASTIC SENSORS TO MEASURE ARM DEFLECTIONS: SYSTEM MODELING & CONTROL

A thesis submitted to the University of Twente in partial fulfillment
of the requirements for the degree of

Bachelor of Science in Electrical Engineering

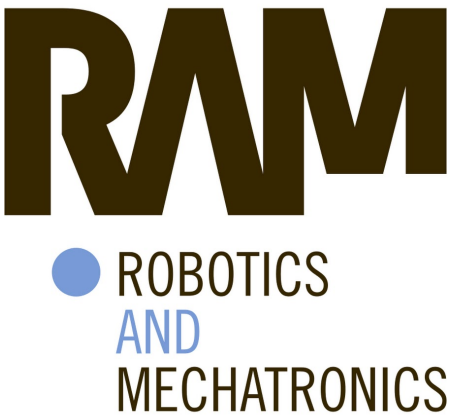
by

Salma Bahaa

July 2022

Salma Bahaa: *Multirotors Embedding Elastic Sensors to Measure Arm Deflections: System Modeling & Control* (2022)

The work in this thesis was made in:



Supervisors: Prof.dr.ir. Antonio Franchi
Dr. Chiara Gabellieri

ABSTRACT

In state-of-the-art UAVs, the input to the control scheme - which affects the position and orientation of the UAV - is considered to be the thrust forces of the propellers. However, the actual input to the control scheme that has any effect on the system is, in fact, the angular velocity of the rotors. A direct relation is drawn between the rotor velocity and the thrust force, which is a very crude estimation that does not allow for the most efficient control of the vehicle in cases like: change in air-density, malfunctions, presence of other objects in the vicinity.

This research builds up on the findings of an alumnus of the University of Twente, E.B. (Bernard) Prakken, who explored the implementation of strain gauges in a multirotor UAV. In his research, these strain gauges are used to estimate the thrust force, which in turn are fed into the control scheme to improve the stability of the system.

Essentially, this report discusses the feasibility of using the thrust forces as control inputs to observe and control the system. The goal to excavate empirical results regarding the observability and controllability from MATLAB and Simulink simulations of the system. The system is first linearized, its observability studied, and then the propeller thrust forces were estimated using an open-loop state estimator. The results showed that the thrust forces can be estimated accurately enough to be used as control inputs to Model Predictive Control scheme, given that the model approximations remain true.

ACKNOWLEDGEMENTS

I would like to thank the Robotics and Mechatronics department for providing me with the opportunity to work on a thesis that I have been passionate about for the longest time, and accommodating my preferences and needs. Foremost, I would like to thank my supervisors, prof.dr.ir. Antonio Franchi and dr. Chiara Gabellieri for their continued support and encouragement throughout this assignment, even when morale was low and times were tough. Dr. Chiara's intelligence, enthusiasm and motivation as my day to day supervisor helped me overcome the obstacles I faced during this time period.

Furthermore, I would like to thank the Ph.D. candidates residing in Carré 3714 for providing me with endless free coffee and tolerating my temper tantrums and existential dread. Special thanks to Yannik Wotte and Christophe van der Walt for lending out a helping hand when the solutions were not always clear. Their insights and patience have definitely helped me on a professional and personal level.

I would also like to thank Nikolai Schauer for being a friend in hard times, a chauffeur when I fractured my ankle, an emotional beacon and a thesis writing buddy.

A special thanks to dr.ir. Dennis Alveringh for being a part of my graduation committee.

At last, I would like to thank Ph.D. candidate Youssef Aboudorra, who helped me with the overarching mathematical topics even though he was not assigned to be my supervisor.

CONTENTS

LIST OF FIGURES

1 INTRODUCTION

1.1 GENERAL OVERVIEW

In recent years, unmanned aerial vehicles (UAVs) have grown in popularity in the scientific community. As control technologies improved and associated costs declined, their use expanded to many applications, such as monitoring and control of natural disasters, photography, and surveillance [1].

In a multirotor (i.e. UAV with more than two rotors), the control scheme is said to take the force produced by each propeller as inputs. The propeller's thrust is, in fact, empirically determined using motor and propeller specifications, thus linking the thrust forces to the propeller's angular velocity and the dynamic model itself. This approach leaves the control scheme vulnerable to external conditions: loss of communication, wind, propeller damage, motor failure, and other factors that impact the actual value of the thrust forces [2].

This thesis will explore the possibility of using embedded elastic sensors (i.e. strain gauges) to precisely estimate the propellers' thrust forces, in an effort to improve the control scheme of a given birotor.

1.2 CONTEXT

Until now, in order to identify the input force to the control scheme of the UAV in question, the spinning velocity of the propeller is used to compute the amount of force. However, this is considered a crude estimation that is necessary, since controlling the spinning velocity instantaneously and accurately is an approximation in and of itself. At this stage, errors in the model itself translate into errors between the commanded control force and the actual input force produced. Adding external disturbances (e.g. air density, wind, walls, etc.) that affect the model, a closed loop controller is preferred.

In this assignment, a state-of-the-art strain gauge is attached to each arm to measure a deflection angle, in an interest to find out what the propeller force is by defining an empirical relationship between those measured deflection angles and the propellers' thrust forces. Note that such gauges introduce elasticity in the system, which influences the model dynamics, as well as system observability and controllability.

Given the aforementioned state of affairs, this research begins by defining clear questions and goals.

1.3 RESEARCH QUESTIONS & GOALS

1.3.1 Questions

This research build up on the findings of an alumnus of the University of Twente, Bernard Prakken, who found an empirical relation between the measurement of the strain gauges and the thrust forces of the propeller. The main research question in this project is **whether or not it is actually feasible to use this empirical relation to**

observe and control a system model of a birotor, taking into account the flexibility introduced by the elastic sensors.

Answering this research question is substantial to the Robotics and Mechatronics group at the University of Twente. In the aerial robotics subgroup, it occurs more often than not that the UAVs are used to perform menial labor and force exertion, in which cases it would be beneficial to know the thrust output of the propellers of the UAV in question, and be able to use that as an input to the control scheme.

1.3.2 Goals

The goals of this research can be discretized into clear bullet points that relate to the overarching topics that are meant to be studied throughout this thesis: system modeling and control.

System Modeling

This topic concerns itself with representing the UAV system in question using a set of mathematical equations (i.e. a mathematical model). By manipulating this mathematical model, the design and analysis of the controller is made possible. Thus, the following goals are defined:

1. Study the observability of the birotor in question.
2. Decouple the dynamic system of linear equation.
3. Estimate the states of the system, given that the propeller forces are also part of the state.

Control

This topic concerns itself with the analysis of a controller: finding the output when the input and the mathematical model are known, and closing the loop such that the system is considered *stable*. Thus, the following goals are defined:

1. Estimate the parameters of the values that govern system behavior, such as the time constant of the first-order relation between the desired PWM and the propeller's actual velocity.
2. Design a controller with the commanded velocity or PWM to the propellers as inputs. Specifically, Model Predictive Control can be used to exploit the new measurements to improve tracking control of the UAV.

1.4 THESIS OUTLINE

Considering that the aforementioned goals are the pillars of this assignment, this thesis paper will have the following construction:

1. **Literature Review**
2. **Modeling**
3. **System Observability**
4. **State Observer**
5. **Controller Design**
6. **Discussion, Conclusion & Recommendations**

2 LITERATURE REVIEW

2.1 INTRODUCTION

A literature review was carried out in order to identify relevant research papers and scholarly sources. Upon reviewing and surveying these papers, an overview of current knowledge on this topic is provided, allowing for the identification of relevant theories, mothods, and gaps in existing research.

In this section, three research papers are evaluated in terms of their relevance to the topic of this report and their most important insights highlighted and discussed.

2.2 DESIGN, MODELING, AND CONTROL OF A NOVEL MULTIROTOR THRUST FORCE SENSOR [2]

Written by an alumnus of the University of Twente, E.B. (Bernard) Pakken, this first paper serves as a general overview of the problem at hand. In the introduction to this paper, a similar problem is described: the rotor velocity is the actual input to the control scheme, albeit the thrust force of the propeller being widely considered as the input. This is because the thrust force is seen to have an effect on the position and orientation of the UAV, which are system states. In this paper, the implementation of strain gauges, that essentially function as sensors, in a multirotor UAV is explored, discussed and analyzed.

The paper makes the following proposition: the strain gauges measure the strain in the arms they are mounted on, that is produced by the thrust of the propeller associated to the arm. Such measurement goes hand in hand with a deflection angle, as previously described in the introduction of this literature review, thus introducing flexibility. This flexibility is, according to this paper, as unwanted effect, since it causes unwanted vibrations and can lead to an uncontrollable system.

Contextually speaking, the paper focuses on the design of the strain gauges and how different setups and environments influence the resulting measurement. However, in chapter 1, the paper talks about something more relevant: the System Modelling of the UAV. To avoid over-complexity, the model should require as few parameters as possible, while still faithfully representing the system. In the model, it is important to include the flexibility in the arms that is introduced by the strain gauges. This leads Pakken to formulate his research goal: can the strain that is measured by the strain gauges be related to the amount of force of the propellers? This research is crucial. Based on the answers to this research question, the control scheme of the UAV at hand can be changed drastically.

Lastly, in section 6.3, the paper tackles an extremely insightful and relevant topic: the Observability of the system. An observability analysis was carried out in order to discuss the relationship between the joint angles (i.e. deflection angles) and the thrust force. Throughout section 3, the author assumes that the sensor is a fixed, rigid body, and uses the EulerBernoulli beam theory to connect the two parameters in question. However, when the UAV is flying, the sensors are not clamped or fixed to anything, due to the absence of ground. Therefore, an empirical research that put the dynamics of the UAV to the test was carried out.

It was expected that the applied force F and the joint angle β to have a linear relationship. While some test trajectories did indeed show a linear relationship, with the peaks and valleys of sinusoidal signals lining up perfectly, (See figures 6.13 and 6.15), one of the trajectories proved that this assumption does not hold. The result in question is shown in figure 6.14. The trajectory demonstrated the UAV moving along the positive side of the x-axis, which means that the UAV has to rotate horizontally. This causes one of the motors to exert a higher force than the other, and vice versa when the UAV stops. Based on these results, the author concluded that applied thrust force can not be measured through linear computations, thus debunking the Euler-Bernoulli model discussed previously. Essentially, this paper acts as a proof of concept: the force can indeed be estimated (under some reservations and conditions) from the angle of deflection that is measured by the strain gauges, based on empirical research that the author generously carried out and provided.

2.3 OPTIMAL KALMAN FILTER FOR STATE ESTIMATION OF A QUADROTOR UAV [3]

After successfully modeling the system and simplifying the system equations, the states of the system need to be estimated. In control theory, state observers or state estimators provide an estimate of the internal state of a given real system (i.e. the multirotos with embedded elastic sensors) [4]. To do so, measurements of the input and output of the real system are required, and obtaining such measurements has been established in the second section of this paper. By combining information about a system's behavior and external measurements to estimate the true state of the system, a state observer can be implemented successfully. As previously discussed, the system in question is linear. A common observer used for linear systems is the Kalman Filter.

This research paper has a main objective of studying the Optimal Kalman Filtering method for estimating the states of a UAV. In control theory, a Kalman Filter (also known as Linear Quadratic Estimator) is an algorithm that uses a series of measurements observed over time, while incorporating internal disturbances, including the white Gaussian process (statistical noise), measurement noises and other inaccuracies. Drawing from probability theory, the estimates that the Kalman filter produces are more accurate in comparison to the estimates produced from one measurement [5]. This, however, implies that a Kalman filter requires sufficient measurements.

The authors also provide the design of a discrete-time flight controller in order to perform high accuracy position and altitude tracking. However, since this paper is only relevant for its Optimal Kalman Filtering, the controller design section of this paper will not be discussed.

The paper starts by introducing the kinematic and dynamic model of the quadrotor. From the basic understanding of Kalman filtering, it is clear that the dynamic characteristics of the quadrotor must be known in order to successfully build the Kalman filter for the state estimation. Therefore, the system is expressed in the form of a discrete-time state model:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(K) + Gw(k) \\ y(k) &= Cx(k) + v(k) \end{aligned}$$

where $x(k)$ is the state vector at time k, A is the state transition matrix, B is the control distribution matrix, G is the transition matrix of system noises, $y(k)$ is the measurement vector at time k, C is the measurement matrix, $w(k)$ and $v(k)$ are the white Gaussian process and measurement noises.

Hereafter, the Kalman filter is found by separating the process into 5 steps:

1. **State prediction:** the Kalman filter produces estimates of the current state vector $x(k)$, along with its associated uncertainties that were previously mentioned.

2. **Covariance prediction:** these estimates are then updated as a new measurement is obtained using a weighted average, where the weights are calculated from the covariances in the system.

3. **Kalman gain:** the estimates that have greater certainty are assigned more weight. This certainty-grading of the current state vector and new measurements can be discussed in terms of the Kalman filter's gain. The Kalman gain is essentially the weight given to the measurements and current state vector, which can be "tuned" to achieve a particular performance.

4. **State estimation:** since the Kalman filter algorithm is recursive, the process is repeated at every discrete time step k to reach an optimum value for the state vector.

5. **Covariance estimation:** the Kalman filter also estimates the covariance matrix, since the new estimate and its covariance inform the Kalman filter about the prediction of the next iteration. As the Kalman filter reaches an optimum state vector, an optimum covariance matrix is also found.

fig. 2.1 shows, graphically, how the Kalman Filter uses the probability density functions of the states (i.e. the covariance) to estimate, or tune, the optimum value for the state vector.

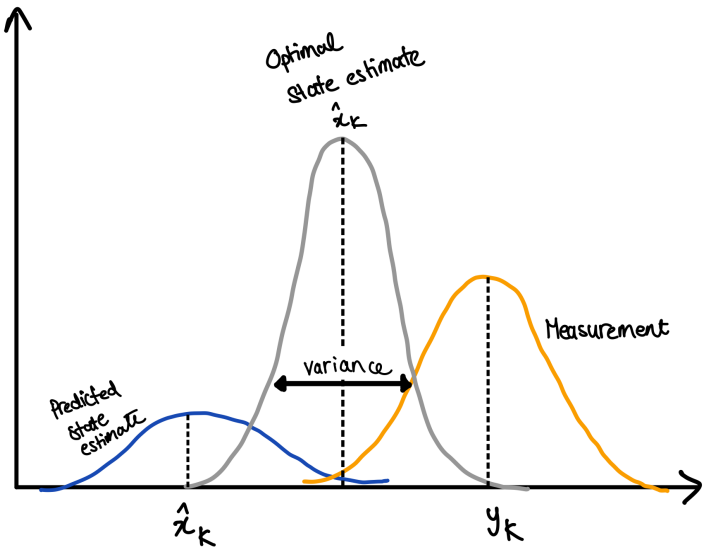


Figure 2.1: How the Optimal Kalman Filter is used to minimize the variance of the predicted state $\hat{x}_k$ to obtain an optimal state estimate.

The paper proceeds to test the Kalman filter in combination with the flight controller. The simulation tests included both process and measurement noise, and the authors reached the following conclusions about the Kalman filter:

> "Under the noise disturbances, the high accuracy position and attitude tracking are obtained using the Optical Kalman Filter. The Optical Kalman Filter is an effective estimation method in autonomous navigation systems, especially its good robustness to measurement noise."

2.4 MODEL PREDICTIVE PATH-FOLLOWING CONTROL OF AN AR.DRONE QUADROTOR [6]

The final stage to be addressed in this literature review is the controller design. This last paper addresses the design and implementation of Model Predictive Control for path-following in a UAV. Model Predictive Control is a novel concept to electrical engineering bachelor students, this paper is deemed very relevant and useful for its comprehensible overview of MPC.

In section 2, the authors write:

> "Model Predictive Control (MPC) is a general designation for controllers that make an explicit use of a model of the plant to obtain the control signal by minimizing an objective function over a time horizon in the future."

In control theory, Model predictive control (MPC) is an advanced method of controller design that is typically used for plants with a certain set of constraints. Such controllers usually rely on linear empirical models, thus rendering MPC suitable for this assignment. The main advantage of MPC is that it predicts the change in the dependent variables (constraints) of the system (that are caused by the changes in the independent variables, such as disturbances). MPC considers an iterative, finitehorizon optimization of a model. By using current system measurements and dynamic states, the MPC calculates the future changes in dependent variables. These changes are calculated to hold the dependent variables close to target while honoring constraints on both independent and dependent variables [7].

2.5 CONCLUSION

In conclusion, all points mentioned in the introduction have been assessed through the means of reviewing relevant, peer-reviewed research papers. Pertinent and applicable insights and information have been extracted, discussed, and critiqued in terms of how helpful they are to the assignment. This literature review may then continue to be a solid ground for this bachelor thesis.

3 MODELING

3.1 INTRODUCTION

In control system theory, a plant or a system can be represented with a set of mathematical equations known as a *mathematical model* [8]. Such models, as previously shed light upon, are useful for the analysis and design of control systems.

In this chapter, the model will be introduced, its nuances explained, linearized and manipulated into a state space form such that its observability and controllability can be studied.

3.2 MATHEMATICAL MODEL

3.2.1 Overview

Considering the birotor in question, a list of variables may be defined to describe the plant, as shown in fig. 3.1.

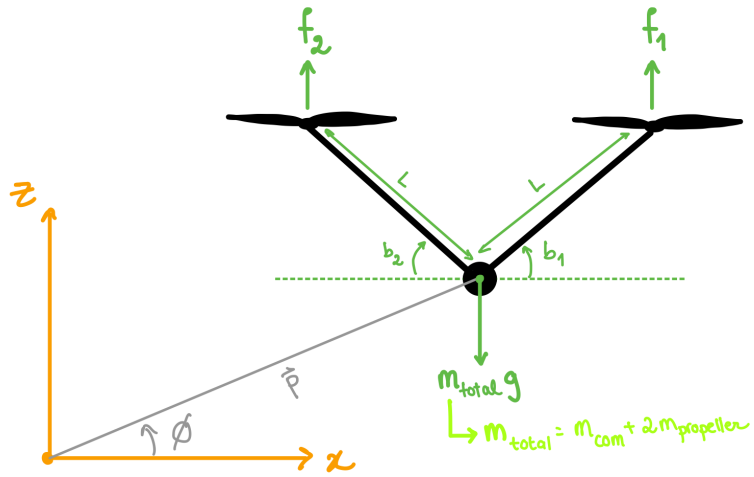


Figure 3.1: A picture of the plant, showing the variables that govern the behavior of the birotor

The angles of deflection b_1, b_2 are measured using the state-of-the-art elastic sensors and are defined in an inertial frame of reference, as well as their derivatives. The same applies to the tilt angle ϕ, then length L of the propeller arm, the mass m of the UAV, and the masses of the propeller arms. The position of the center of mass of the birotor, velocity and acceleration (i.e. the first and second derivatives of the position, respectively) of the birotor are also defined in the inertial frame of reference and are included in the system configurations. The birotor is essentially a projection of a quadrotor on a plane; the position is then defined as:

$$p = \begin{bmatrix} x \\ z \end{bmatrix} \tag{3.1}$$

The system configurations are defined as follows:

$$q = \begin{bmatrix} p \\ \phi \\ b_1 \\ b_2 \end{bmatrix}, \dot{q} = \begin{bmatrix} \dot{p} \\ \dot{\phi} \\ \dot{b_1} \\ \dot{b_2} \end{bmatrix}, \ddot{q} = \begin{bmatrix} \ddot{p} \\ \ddot{\phi} \\ \ddot{b_1} \\ \ddot{b_2} \end{bmatrix} \tag{3.2}$$

The propeller positions are defined in terms of the position p of the center of mass of the birotor, the length of the propeller, the angle of deflection and the tilt angle ϕ. This transformation is done by the means of a *rotation matrix* to go to the body-fixed frame of reference:

$$\mathbf{R} = \begin{bmatrix} cos(\phi) & -sin(\phi) \\ sin(\phi) & cos(\phi) \end{bmatrix} \tag{3.3}$$

This matrix is typically used to rotate points in a 2-dimensional plane counter-clockwise through an angle ϕ with respect to the original position p [9]:

$$p_{propeller} = p + \mathbf{R}(\phi)(\mathbf{R}(b_{1,2})L) \tag{3.4}$$

The velocities of the propellers are also defined by taking the *jacobian* of eq. (3.4). These velocities are important in the calculation of the kinetic energy of the birotor to be substituted in the *Euler-Lagrangian dynamic equation*.

3.2.2 Euler-Lagrangian Dynamic Equations

In theory of robotics, the Euler-Lagrangian equations are an energy-based method that is used to represent dynamic equations in a symbolic form [10]. This method is typically used to to study the dynamic properties of systems and analyze control schemes. A general assumption that all N links in the model are *rigid bodies* [11].

Considering eq. (3.2), the Lagrangian function is written as follows [12]:

$$L(q, \dot{q}) = K(q, \dot{q}) - U(q) \tag{3.5}$$

Where T and U are the kinetic and potential energies of the model, respectively. By assuming the stationary-action principle[1] as well as the principle of virtual work[2], the Euler-Lagrangian equations can then be written as:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = \tau_i \tag{3.6}$$

Where τ_i is defined as the non-conservative generalized forces, for $i = 1, ..., N$ [11].

The kinematics of the UAV are calculated by first finding the potential and kinetic energies, as dictated by eq. (3.6). The kinetic energy is calculated while taking into consideration the mass of the birotor, the mass of the propellers, the angular velocity $\dot{\phi}$, the velocity of the birotor, and the velocities of the propellers, to fulfill the equation:

$$K = \frac{1}{2}mv^2 \tag{3.7}$$

1 The stationary-action principle is a variational principle that produces the equations of motion of a mechanical system when applied to that system [13]

2 The principle of virtual work states that in equilibrium the virtual work of the forces applied to a system is zero [14]

The potential term U includes both the gravitational potential energy and the elastic potential energy that is introduced by the flexibility of the strain gauges, both fulfilling the equations:

$$U_{gravitational} = mgh \tag{3.8}$$

$$U_{elastic} = \frac{1}{2}kx^2 \tag{3.9}$$

Where g is the gravitational acceleration of approximately $9.81ms^2$, and k is the elastic spring constant of the strain gauges (refer back to Prakken's paper). After finding K and U, eq. (3.5) can then be found and substituted in eq. (3.6).

3.2.3 State of the Art

The non-conservative generalized forces found in eq. (3.6) can be expressed in terms of inertia, coriolis and gravitational forces:

$$M(q)\ddot{q} + C(q,\dot{q}) + G(q) = \tau \tag{3.10}$$

To find an expression for all system configurations $\ddot{q}$, M, C and G have to be found first before rearranging eq. (3.10) to change the subject to $\ddot{q}$. Essentially, eq. (3.6) is expanded into a linear inertial terms M that is dependant on the acceleration $\ddot{q}$, quadratic centrifugal and coriolis terms C that are dependant on the velocity $\dot{q}$, and nonlinear gravity terms that are dependent on the configuration q [11].

$$M = \mathbf{J}(\tau, \ddot{q}) \tag{3.11}$$

$$G = \begin{bmatrix} 0 \\ g(m+2m_p) \\ -Lgm_p(cos(b_1+\phi) - cos(b_2+\phi)) \\ b_1k - Lgm_pcos(b_1+\phi) \\ b_2k + Lgm_pcos(b_2+\phi) \end{bmatrix} \tag{3.12}$$

$$C = \tau - M - G \tag{3.13}$$

Where M is the jacobian of the non-conservative forces τ with respect to $\ddot{q}$, and G is found by exploring the dependency of each entry in the system configuration q on gravity [15].

The most recent stage in the development of this model assigns symbolic values to the all system configurations $\ddot{q}$. The following equation fully describes the actuated dynamics of the birotor in question.

$$\ddot{q} = M^{-1}(-C - G + \tau) \tag{3.14}$$

Using MATLAB's Symbolic Math Toolbox [16], $\ddot{q}$ can be represented, manipulated and solved symbolically, without having to assign values to all of the aforementioned variables. This allows for the analysis to take on a more comprehensive approach, where the states, their trajectories, and the controller behaviour can be clearly observed.

3.3 SYSTEM LINEARIZATION

3.3.1 Overview

The birotor in question, much like most real systems, exhibits nonlinear behavior, where its inputs and outputs can be expressed in the form

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

Typically, it is an arduous process to analyze nonlinear systems and use them in controller design [17]. Therefore, nonlinear systems are linearized to obtain a linear model that is much easier to deal with.

Given the current system equations, a few steps need to be taken before the system can be approximated linearly to a state space model.

3.3.2 Change of Coordinates

In order to simplify the system equations, consider the sum of and difference between the two deflection angles, where the sum $b_s = b_1 + b_2$ and the difference $b_d = b_1 - b_2$. This simple substitution allows for $\ddot{q}$ to be simplified into simpler terms, and will allow for the linearization process to be easier as well, as will be explained later.

In order to introduce these variables, the system equations need to be rewritten in terms of those new variables. This means that the system equation needs to be rewritten from

$$q = \begin{bmatrix} p \\ \phi \\ b_s \\ b_d \end{bmatrix}, \ \dot{q} = \begin{bmatrix} \dot{p} \\ \dot{\phi} \\ \dot{b_s} \\ \dot{b_d} \end{bmatrix}, \ \ddot{q} = \begin{bmatrix} \ddot{p} \\ \ddot{\phi} \\ \ddot{b_s} \\ \ddot{b_d} \end{bmatrix} \tag{3.15}$$

Starting with q: by changing the subjects, the deflection angles can be substituted in the system equations with:

$$b_1 = \frac{b_s + b_d}{2} \tag{3.16}$$

$$b_2 = \frac{b_s - b_d}{2} \tag{3.17}$$

Moving on to $\dot{q}$: the derivative terms $\dot{b_s}$ and $\dot{b_d}$ can be substituted into $\dot{q}$, by simply differentiating eq. (3.16) and eq. (3.17), and carrying out the following substitution:

$$\dot{b_1} = \frac{\dot{b_s} + \dot{b_d}}{2} \tag{3.18}$$

$$\dot{b_2} = \frac{\dot{b_s} - \dot{b_d}}{2} \tag{3.19}$$

Finalizing with $\ddot{q}$: in order to change the coordinates for $\ddot{q}$, the second derivative terms $\ddot{b_s}$ and $\ddot{b_d}$ have to first be found out.

Consider the original $\ddot{q}$ in eq. (3.2). The aforementioned derivative terms can simply be found out via some matrix manipulation:

$$\ddot{b_s} = \ddot{b_1} + \ddot{b_2} = \ddot{q}_4 + \ddot{q}_5 \tag{3.20}$$

$$\ddot{b_d} = \ddot{b_1} - \ddot{b_2} = \ddot{q}_4 - \ddot{q}_5 \tag{3.21}$$

where $\ddot{q}_4$ and $\ddot{q}_5$ denote the 4^{th} and 5^{th} entries in $\ddot{q}$ respectively.

Now that all matrices are written in terms of b_s, b_d, and their derivative terms and all required substitutions have been carried out successfully in the system equations, the change of coordinates is complete.

3.3.3 Computing Equilibrium

The key idea to linearizing the system is to create a linear approximation of the nonlinear system that is valid in a small region around an operating point, known as the *equilibrium point*. At this point, all states are constant (i.e. their derivative terms are equal to zero) [17].

Mathematically, a system that is described by a differential equation $\dot{x} = f(x, u)$ has an equilibrium point (x_e, u_e) if $f(x_e, u_e) = 0$.

In order to solve for the above equation, the system is redefined as follows:

$$x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}, \ \dot{x} = \begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} \tag{3.22}$$

By rewriting the system equations as shown in eq. (3.22), the equations to be solved are clearer: to find (x_e, u_e), the derivative terms $\dot{p}$, $\dot{\phi}$, $\dot{b_s}$ and $\dot{b_d}$ are zeroed, and then $\dot{x}$ has to be equated to zero.

Using MATLAB, the equations to be solves are as follows:

$$\frac{b_d k(b_s m + b_s m_p + m\phi)}{Lm(m + 2m_p)} == 0 \tag{3.23}$$

$$-\frac{b_d k + Lgm}{Lm} == 0 \tag{3.24}$$

$$\frac{b_s k}{J} == 0 \tag{3.25}$$

$$-\frac{JLf_1 - JLf_2 + Jb_s k + 2L^2 b_s k m_p}{JL^2 m_p} == 0 \tag{3.26}$$

$$-\frac{Lf_1 m + Lf_2 m + b_d k m + 2b_d k m_p}{L^2 m m_p} == 0 \tag{3.27}$$

These equations were then solved manually to maintain the integrity of the calculation.

- By taking a look at the equations above, it can be noticed that the position p does not show up in the equations. This is because the position of the UAV in the inertial frame of reference does not influence its equilibrium and stability in any way, and can therefore be left as symbolized variables in x_e.
- The tilt angle ϕ was set to zero, since the model assumes *near hovering conditions* of the UAV at static equilibrium. The model also assumes *small angle approximations*.
- From eq. (3.25), $b_s = 0$ at equilibrium. This makes sense; the deflection angles are equal and opposite to one another (refer back to fig. 3.1).
- From eq. (3.24), at equilibrium:

  $$b_d = -\frac{Lgm}{k}$$

  When b_d is then substituted back into the expression for position x, the gravity term g will appear. This makes sense, because the birotor performs the underactuated dynamics of a quadrotor, yet cannot translate in the x-direction. The birotor has to first tilt, and then move in the x-direction, and this tilting is dependant on g.

- To find u[3], eq. (3.26) and eq. (3.27) were solved simultaneously:

$$f_1 = f_2 = \frac{1}{2}g(m + 2m_p)$$

This, again, makes sense that the propellers will exert equal forces at equilibrium, to counteract the gravitational force.

After finding the equilibrium point, the system may then be linearized around this point.

3.3.4 Jacobian Linearization to State Space Representation

A linearized model can be fully expressed by state space equations in the form:

$$\dot{x} = Ax + Bu \tag{3.28}$$
$$y = Cx + Du \tag{3.29}$$

where eq. (3.28) is the dynamic equation in terms of the state matrix A and input matrix B, and eq. (3.29) is the output equation of the system in terms of the output matrix C and feed-through matrix D.

This section will explore the expression of eq. (3.28).

The linearization is based on *Taylor approximations* [18], where:

$$\begin{aligned}\dot{x} = f(x,u) &= f(x_e + \delta x, u_e + \delta u) \\ &\approx f(x_e, u_e) + \frac{\partial f}{\partial x}|_{x_e,u_e}\delta x + \frac{\partial f}{\partial u}|_{x_e,u_e}\delta u\end{aligned} \tag{3.30}$$

By relating eq. (3.30) to the dynamic equation eq. (3.28), it can be concluded that:

$$A = \frac{\partial f}{\partial x}|_{x_e,u_e} \tag{3.31}$$
$$B = \frac{\partial f}{\partial u}|_{x_e,u_e} \tag{3.32}$$

Matrix A is then known as the Jacobian of f with respect to x, essentially representing the first differential of f at every point where f is differentiable over small variations of x around the equilibrium point (x_e, u_e). Matrix B is, in turn, the Jacobian of f with respect to u, and represents the first differential of f at every point where f is differentiable over small variations of u around the equilibrium point (x_e, u_e) [19].

In MATLAB, this process is relatively easy, as can be seen in appendix A.1, thanks to the inbuilt `jacobian` function [20]. The following matrices were found:

3 u is the input, which can be expressed with the propeller thrust forces, where $u = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$.

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -g & -g\frac{m+m_p}{m+2m_p} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{k}{Lm} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{k}{J} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -k\frac{2m_pL^2+J}{JL^2m_p} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -k\frac{m+2m_p}{L^2mm_p} & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.33}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -\frac{1}{Lm_p} & \frac{1}{Lm_p} \\ -\frac{1}{Lm_p} & -\frac{1}{Lm_p} \end{bmatrix} \tag{3.34}$$

These matrices can then be plugged back into eq. (3.28) to provide a full representation of the aerodynamic system. In this representation, all state derivatives are considered with respect to their dependencies on the state vector and the input propeller's thrust forces.

Some important observations that can be made about the state equation are:

- The first 5 entries, which concern the first derivatives of the state, show that the homogeneity of the equation is preserved: the velocities are not influenced by the the force inputs and are only dependent on themselves.
- As previously explained, the acceleration in the x direction shows dependencies on gravity, given that g appears in the 6^{th} entry in matrix A, which confirms the speculations made about the under-actuated dynamics of the birotor.
- By looking at matrix B, the only non-zero entries are the last two row entries. This means that the force inputs only influence the angular accelerations of the small angle deflections b_s and b_d.

The output dynamics in eq. (3.29) will be derived after what is known as *state augmentation*, explained in section 4.2.

4 SYSTEM OBSERVABILITY

4.1 INTRODUCTION

In control system theory, the system must be deemed *observable* in order to see the processes that are taking place inside the system under observation. In essence, observability is the ability to measure the internal states of a system by examining its outputs, namely sensor data [21].

One of the main goals of this research is to determine whether the propeller thrust forces can be estimated and are observable by relying on the measured propeller speed output. In order to verify this, the states of the system have to be *augmented*, such that the propeller forces are considered to be part of the states. Then, the observability can be studied depending on which outputs are selected as observable.

4.2 AUGMENTING THE STATES

In order to find the state space representation of the output that is necessary to study the observability of the system, the states need to be augmented first. This means that the propeller thrust force will be considered as part of the state vector $\tilde{x}$. In this way, the observability analysis will be more meaningful. Therefore, the state vector can be rewritten as follows:

$$\tilde{x} = \begin{bmatrix} q \\ \dot{q} \\ f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} x \\ u \end{bmatrix} \tag{4.1}$$

since the thrust propeller forces are lumped together in the input vector u.

Therefore, considering eq. (4.1), the augmented state space representation can be written as follows:

$$\begin{aligned} \dot{\tilde{x}} = \begin{bmatrix} \dot{x} \\ \dot{u} \end{bmatrix} &= \tilde{A}\tilde{x} + \tilde{B}\tilde{u} \\ &= \begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix} \tilde{x} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tilde{u} \end{aligned} \tag{4.2}$$

where $\tilde{u}$ is selected to include the first derivatives of the thrust forces, $\dot{f}_1$ and $\dot{f}_2$ to maintain the homogeneity of the equation.

The output eq. (3.29) will also be augmented, where the augmented output matrix $\tilde{C}$ "selects" which outputs are of interest with regards to estimating the propeller thrust forces.

To properly address the main goal of this research, the deflection angles will be selected, since the whole idea is to explore the feasibility of estimating the thrust forces via the deflection angles measured by the strain gauges attached. Therefore, b_s, b_d, $\dot{b_s}$ and $\dot{b_d}$ will be selected by $\tilde{C}$. Intuitively speaking, the position p and the the tilt angle ϕ, as well as their first derivatives, need to be selected in the output matrix such that the plant is observable. Considering that $\tilde{C}$ is a row matrix, 1's can be inserted in correspondence to the selected states. Therefore:

$$\tilde{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{4.3}$$

where the 1's correspond to x, z, ϕ, b_s, b_d, $\dot{x}$, $\dot{z}$, $\dot{\phi}$, $\dot{b_s}$ and $\dot{b_d}$ in order, and the 0's correspond to f_1 and f_2.

In this application, the feed-through matrix $\tilde{D}$ is zero. As long as the mathematical model of the birotor is understood and accurately captured, which has already been established, matrix $\tilde{D}$ can provide a performance boost that can mitigate some unwanted influences of controllers, e.g. a PID controller. Therefore, in this case, matrix $\tilde{D}$ is set to zero, and further exploration of what this matrix could be is done after the controller design.

The output eq. (3.29) can then be written as follows:

$$\tilde{y} = \tilde{C}\tilde{x} \tag{4.4}$$

4.3 OBSERVABILITY

4.3.1 Overview

Now that the system is modeled in state space representation, its observability can be studied using conventional methods, such as the observability matrix. This section will provide a more formal and full definition of observability, adding onto section 4.1.

Circling back to the original research question, it is important to find out whether the forces can be estimated. After augmenting the states and considering the forces to be state variables, some limitations are introduced to the calculation of the observability of the system. Considering the forces as state variables allows us to explore whether their state trajectories can be estimated by measuring the outputs of the strain gauges.

4.3.2 The Matrix

Drawing from this definition, the observability matrix makes use of the output matrix and the state matrix to determine how many observable states exist in a system. Formally, the observability matrix is calculated as follows:

$$\mathcal{O} = \begin{bmatrix} C \\ CA \\ CA^2 \\ . \\ . \\ . \\ CA^{n-1} \end{bmatrix} \tag{4.5}$$

where n is the number of state variables [22]. In this case, the augmented $\tilde{C}$ and $\tilde{A}$ are used in the calculation of the observability matrix.

The *rank* of the observability matrix, as calculated in eq. (6.1), is the measure that is sought out. In linear algebra, the rank of a matrix is defined as the vector space that is spanned by its columns. Ideally, the rank of the observability matrix should be equal to the number of states in the state vector. If that is the case, then the system is fully observable.

In MATLAB, the obsv function that calculates the observability of a given system does not accept symbolic input arguments [23]. Therefore, a new function was written, found in appendix A.2 to calculate the observability of the system.

The observability matrix of this system has *full rank*, which means that all states are observable [22]. Note that matrix $\tilde{C}$ selects all possible hypothetical sensors, which is inefficient and not realizable. The next step is to select as few states as possible, in turn utilizing fewer sensors, that allow the observability matrix to sustain its full rank.

4.3.3 First Results

The matrix $\tilde{C}$ shown in eq. (4.3) was adjusted to select as few states as possible, such that the observability matrix $\mathcal{O}$, shown in eq. (6.1) to maintain its full rank.

When only the states b_s and b_d are selected, the rank of the observability matrix turns out to be 6. This means that, out of the 12 states, 6 of them are observable. The null space of $\mathcal{O}$ holds the information that indicates which states these are, and is usually referred to as the *unobservable subspace* [24]. Using MATLAB function null() [25], the unobservable subspace is:

$$null = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.6}$$

where the 1's correspond to the unobservable states. Considering the state vector is eq. (4.1), it can be seen that the unobservable states are x, z, ϕ, $\dot{x}$, $\dot{p}$, $\dot{\phi}$. **This means that, by only measuring the deflection angles, then the deflection angles themselves are observable, their derivatives, as well as the thrust forces.**

After reaching this resounding result, matrix $\tilde{C}$ was changed to see how utilizing different sensors affects the observability of the system. The following observations were made:

- If the positions x and z are not explicitly selected, they will remain unobservable, even if their derivatives $\dot{x}$ and $\dot{z}$ are selected.

- If the output matrix $\tilde{C}$ selects *only* the positions x and z, then the observability matrix $\mathcal{O}$ has full rank, and the system is fully observable.

4.4 CONCLUSION

4.4.1 Main Insights

In chapter 4, the first results of this research were found: the thrust forces of the propellers can indeed be estimated through the strain gauge measurements, i.e. the small angle deflections of the arms.

By further examining how different measurements influence the observability of the system, it was found out that measuring the position provides complete observability of the system. This is not strange, since, at the end of the day, a linear system can be considered a chain of integrators, and the output is measured at the end of this chain.

4.4.2 Model Reliance

This result, however, shows that the system is currently reliant on the mathematical model itself, as described in section 3.2. This might be an indicator that the approximations that were made to obtain a linear model of the birotor were, in fact, not true to the real physical system. It could be argued that some information about the real physical system was lost along the way, which gives counterintuitive results while exploring the observability of the system.

This could potentially be solved by relaxing some of the assumptions and approximations in the model, or by introducing an additional state to the system that represents the uncertainty. In this way, the observability analyses could be conducted in a more meaningful way.

However, the estimation of the forces from the position could be very slow, prone to noise, and not robust to model uncertainties. So, it is still interesting to see of the forces are observable from the angles.

4.4.3 Back to Basics

These insights and conclusions beg the question: is the original research goal properly formulated?

Given a multirotor, it could be considered a rigid body, to which forces are applied such that it hovers, translates, and rotates. In order to be able to control an input, e.g. these applied forces, they have to be assigned *precisely*. Usually, it is assumed that there is a linear map between the spinning velocity of the propellers and the thrust forces of the propellers, and it is therefore sufficient to use the spinning velocities as control inputs[1]. However, the existence of this linear, fixed map is untrue. It is not possible to assign the forces with such great precision as to control them, which is the main premise of this thesis. The main function of the strain gauges is to be able to estimate the trajectories of the thrust forces, so that the control loop can be closed on the forces as inputs.

The strain gauges introduce *flexibility* in the system, as previously described in section 3.2. The observability analysis in section 4.1 should answer the question: **is it feasible to estimate the propeller thrust forces using the measurements from the stain gauges?**

The results that are discussed above show that the position of the rotor can be used to estimate the forces, which implies that the current system is heavily reliant on the model, and is very sensitive to uncertainties and external disturbances.

The point of "closing the control loop" and using feedback, is to cope with the uncertainty of the parameters.

1 There is a *cascading controller* effect, since there is a controller (considered a black box in this thesis) that assigns the spinning velocity to the propeller as requested.

5 STATE ESTIMATION

5.1 INTRODUCTION

Currently, the system has been deemed fully observable, albeit being very sensitive to external disturbances. The next step is to implement a *state observer*. In control theory, a state observer is a dynamical system that produces estimates of the state variables of the system under consideration. In order to close a feedback loop to control and stabilize a system, all of its state space variables must be available at all times [21][26]. This, intuitively, is not feasible. It is usually not practical to measure all state variables, since the added sensors will also add weight, complex modeling dynamic, and are costly. Therefore, by using a state observer, the system can be reconstructed from the available measurements.

In this chapter, a Simulink model has been provided, based on the mathematical model described in section 3.2. The goal is to estimate the forces, and compare these results to what is produced by the empirical relation defined by Prakken in section 2.2. This comparison will allow us to draw conclusions whether the estimation is adequate enough to control and stabilize the system. An overview of the model is shown in appendix B.1. This model was developed by Ph.D. candidate Youssef Aboudorra, for the master's course *Control for UAVs*, and further adapted by alumnus Bernard Prakken to suit the specific birotor in question by defining a separate MATLAB file parameters.m that includes all birotor specifications. This file gets called by Simulink when the model is opened.

5.2 METHODOLOGY

5.2.1 Force Estimation

Overview

By looking at the original state equation eq. (3.28), given that the input $u = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$, the forces can be estimated via linear model inversion. What this means is that, by looking at the state derivatives and their dependencies on the state variables as well as the forces, it is possible to re-arrange the dynamic equation to make u the subject, and estimate the forces using the available states and state derivatives from the measurements.

To elaborate, consider the last two entries in the dynamic equation (which are the only two entries that are influenced by the input forces, see section 3.3.4), taking into consideration eq. (3.33) and eq. (3.34):

$$\begin{bmatrix} \ddot{b}_s \\ \ddot{b}_d \end{bmatrix} = \begin{bmatrix} -k\frac{2m_pL^2+J}{JL^2m_p} & 0 \\ 0 & -k\frac{m_2m_p}{L^2mm_p} \end{bmatrix} \begin{bmatrix} \dot{b}_s \\ \dot{b}_d \end{bmatrix} + \begin{bmatrix} -\frac{1}{Lm_p} & \frac{1}{Lm_p} \\ -\frac{1}{Lm_p} & -\frac{1}{Lm_p} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} \tag{5.1}$$

Given that the angular velocities and accelerations of b_1 and b_2 are known from the model, eq. (5.1) can be rearranged as follows:

$$\begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} -\frac{1}{Lm_p} & \frac{1}{Lm_p} \\ -\frac{1}{Lm_p} & -\frac{1}{Lm_p} \end{bmatrix}^{-1} \left(\begin{bmatrix} \ddot{b_s} \\ \ddot{b_d} \end{bmatrix} - \begin{bmatrix} -k\frac{2m_pL^2+J}{JL^2m_p} & 0 \\ 0 & -k\frac{m_2m_p}{L^2mm_p} \end{bmatrix} \begin{bmatrix} \dot{b_s} \\ \dot{b_d} \end{bmatrix} \right) \tag{5.2}$$

Implementation

This is considered an open-loop estimator, and is a concrete first step to affirm that the forces can indeed be estimated, given that the model closely resembles the birotor. To test this in Simulink, a new MATLAB function block was created to take in $\dot{b_s}$, $\dot{b_d}$, $\ddot{b_s}$, $\ddot{b_d}$, and calculates the forces using the matrix inversion according to eq. (5.2). It is important to note that all the aforementioned inputs are assumed to be readily available from the sensors on the birotor, i.e. the strain gauges, and can be directly used from the model.

Since the model is non-linear, whereas matrix inversion is drawn up from the linearized model that operates around an equilibrium point, it is important to define that operating equilibrium point for the inputs to the MATLAB block. As was explained in section 3.3.3, given that all the inputs are first and second derivatives, their equilibrium is defined as 0. The operating point for the forces, previously calculated, is found by running the nonlinear Simulink model and noting down the steady-state value of the forces. The value was found to be $22.4465N$. This MATLAB function can be found in appendix A.3

The output can then be plotted and compared to the forces produced by the non-linear model. A trajectory planning block is defined, such that a *reference trajectory* can be fed to the UAV model (including a position and a rotation trajectory). This allows us to explore what the UAV would do under different circumstances, thus allowing us to draw more concrete conclusions.

5.3 SIMULATIONS & RESULTS

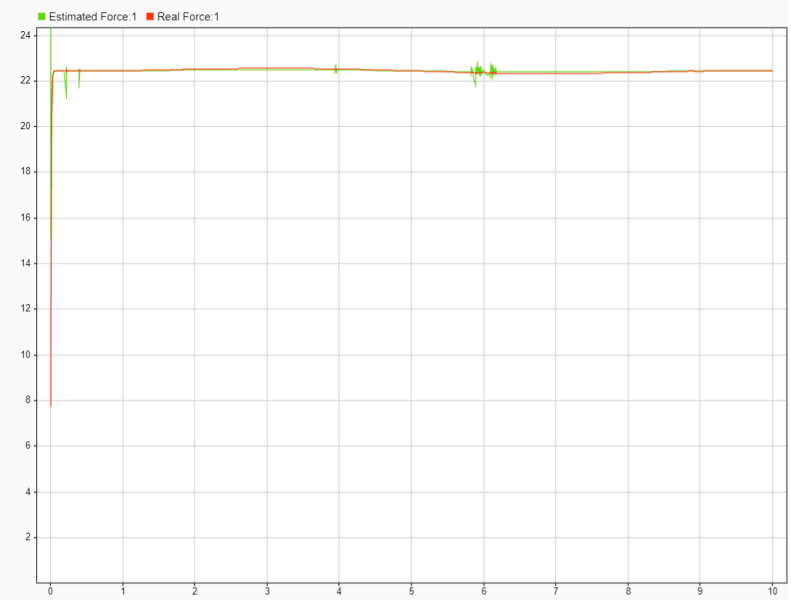
5.3.1 Overview

In Simulink, a trajectory planning block is defined such that it is possible to explore the UAV's behavior when subjected to different reference trajectories - at near-hovering conditions, as well as trajectories that vary from gentle maneuvers to more aggressive ones. The trajectory planning block consists of a position trajectory and a rotation trajectory in 3D space. The initial conditions are first defined, then the trajectory is set by defining the end conditions of the trajectory. The MATLAB code for the trajectory planning can be found in appendix A.4.

5.3.2 Firing Up the Simulator

Since the linearized model, and in turn the estimation of the forces, is based on the assumption that the UAV is in near-hovering position, the position trajectory was set such that the UAV does not move from its initial position or rotate. Given that the two forces are identical, and so are their estimated counterparts, only one of the two forces is plotted in fig. 5.1, along with its estimation.

More tests were conducted to see how the estimator behaves as the UAV moves. The reference rotation trajectory was kept the same, but the UAV was instructed to move from $p = (0, 0, 0)$ to $p = (0.5, 0, 0.5)$ and $p = (1, 0, 1)$ (respectively, in two separate tests) in the xz-plane, thus maintaining the homogeneity of the 2D model. The position trajectory is set to follow a smooth, quintic polynomial function, such that the UAV does not undergo any aggressive maneuvers that would be considered far from the operating point. The results to these simulations can be found in appendix C.1, and are consistent with the results displayed in fig. 5.1.



(a) Full-scale.

(b) Close-up.

Figure 5.1: The force (N) - time (s) graphs of the estimated force (in green) and the real force (in red), given a static trajectory.

5.4 DISCUSSION

5.4.1 General Remarks

From the previous plots, it can be seen that the force estimator tracks the real forces with minimal visible error. Some general remarks as summarized below:

- The estimator takes a number of samples to start tracking the forces. This is because of how Simulink initializes the force estimator.

- The force estimator is sensitive to noise. If the real forces show any small fluctuations, the estimator output amplifies these fluctuations, before tracking the real forces closely again. This sensitivity to noise implies that the estimator

is reliant on how accurate the linearized model is, making the force estimator vulnerable to external disturbances.

- The error between the real and estimated forces seems to fluctuate in a sinusoidal waveform similar to the steady-state waveform of the forces. However, these fluctuations remain between $\pm$ 0.003, and the larger spikes remain between $\pm$ 0.004 - 0.014, with anomalies that can reach 0.029. The results remained consistent throughout all 3 tests (see fig. 5.2 and compare with the figures in appendix C.2). This is a mean absolute error of $0.017N$, which is negligible compared to the actual quantities of the forces. The calculation of the mean absolute error is shown in appendix A.6.

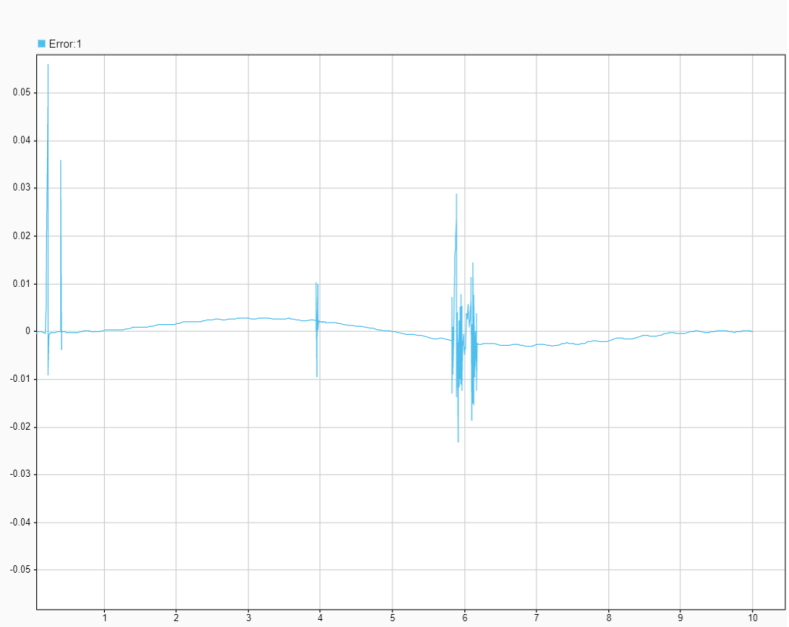


Figure 5.2: Normalized error plot of $f_{real} - f_{est.}$.

5.4.2 The Luenberger Observer

Overview

In control theory, a state observer is often dubbed as a Luenberger observer, crediting David Luenberger, who first introduced the methods used to construct closed-loop state observers today [27]. This section will explore how a Luenberger observer can compensate for what the open-loop force estimator lacks and provide more robustness against noise and external disturbances.

The Math

A Luenberger observer aims to minimize the error dynamics between the measured and estimated states by closing the loop with a gain K that converges the error o [28]. Such feedback loop is visualized in fig. 5.3.

Mathematically, by defining:

$$e_{obs} = x - \hat{x} \tag{5.3}$$

the dynamics of the state observer model can then be expressed as follows:

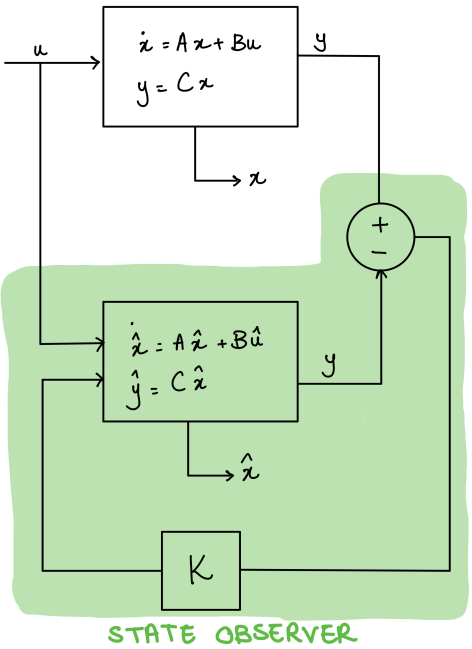


Figure 5.3: A block diagram of a state observer, showing the plant, the estimates of the mathematical model, and how by using negative feedback and adjusting the gain K, the error can be minimized such that x and $\hat{x}$ match.

$$\dot{x} = Ax + Bu \tag{5.4}$$
$$\dot{\hat{x}} = A\hat{x} + Bu + K(y - \hat{y}) \tag{5.5}$$

Given that $y = Cx$ and $\hat{y} = C\hat{x}$, eq. (5.5) can be subtracted from eq. (5.4) to obtain the error dynamics by substituting eq. (5.3):

$$\dot{x} - \dot{\hat{x}} = Ax - A\hat{x} + Bu - Bu - K(y - \hat{y}) \tag{5.6}$$
$$\dot{e}_{obs} = (A - KC)e_{obs} \tag{5.7}$$

The solution to eq. (5.7), given that $y - \hat{y} = Ce_{obs}$, is an exponential function in the form of:

$$e_{obs}(t) = e^{(A-KC)t}e_{obs}(0) \tag{5.8}$$

It is clear that eq. (5.8) is an exponentially decaying function as time approaches infinity if $A - KC < 0$. What this means is that, in state space representation, the poles of $A - KC$ term can be changed by varying K, such that the error dynamics converge [29]. In the case at hand, the system is fully observable to begin with, and the states can be estimated only using matrix A. The significance of having the feedback loop is that varying the gain K will control the decay rate of the error function, alleviating the reliance on the mathematical model and relieving the state observer of some uncertainties and sensitivities.

Implementation

The state observer was implemented in Simulink according to the block diagram shown in fig. 5.4. The Simulink implementation can be found in appendix B.2.

The gain matrix K was first initialized as the identity matrix. The poles of $A - KC$, given that $K = I$, had negative and zero real parts. This means that the feedback

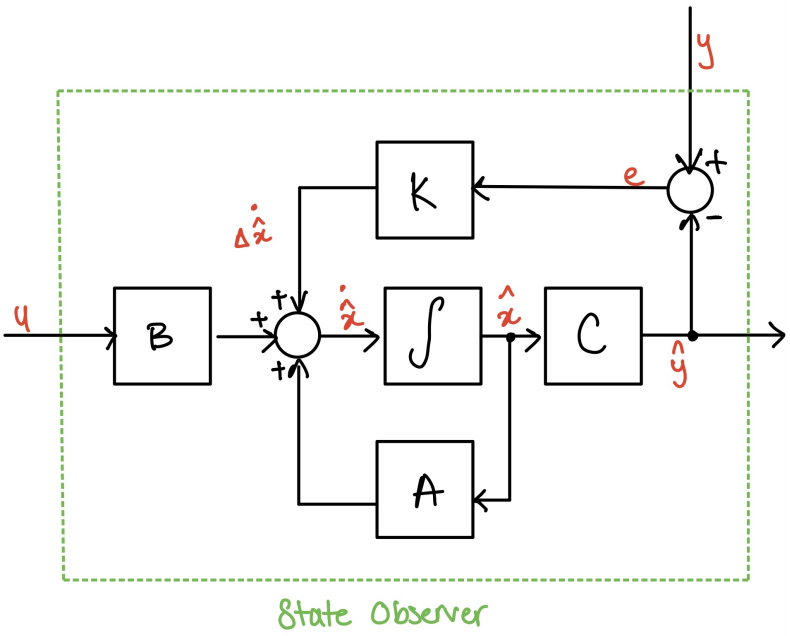


Figure 5.4: A block diagram showing how a state observer reconstructs the states by relying on the state-space model.

loop is *marginally stable* for some states. It was decided to test the state observer with an identity gain matrix first, to explore how the marginally stable states affected the overall performance of the state observer. However, the simulations were not successful due to a number of errors, including:

- MATLAB and Simulink had problems converting the original symbolic matrices to numerical ones, which resulted in a number miscalculations.
- The integral block shown in fig. 5.4 did not function properly due to a size mismatch between matrix dimensions because of how Simulink initializes the feedback loop from x to $\dot{x}$.
- Errors in the state observer caused Simulink to propagate these errors back to the original birotor model, causing the birotor model to behave unexpectedly and incorrectly.
- Simulink would often resolve the errors in the state observer block by entirely bypassing the state observer and producing seemingly perfect results, since the states were replicated exactly.

5.5 CONCLUSION

Even though the state observer was not an overwhelming success, the implemented Simulink model and the observations and conclusions made about the observer provide a good basis for further development. The open-loop estimator on the other hand, did prove that the propellers' thrust forces can indeed be estimated accurately enough to be used as control inputs. This is still a profound result that allows this thesis to move forward.

6 CONTROLLER DESIGN

6.1 INTRODUCTION

Controller design often refers to the techniques used to control the modes of a given dynamical system, and is a realm of knowledge, possibilities, and innovation [21]. Control algorithms such as PID, linear-quadratic regulator, fuzzy-based controllers[1], robust control, and model-predictive control all have the same objective: to govern the system behavior by driving its inputs to a desired state. Different algorithms achieve different levels of optimality and control stability, depending on how well they minimize delay, overshoot, and steady-state error.

In this chapter, the controllability of the UAV system will be studied, and a Model Predictive Control algorithm will be explored. This chapter will not include the implementation of the controller and will not present any results to the feasibility of implementation, since controller design is a complicated process, with the chosen control algorithm adding to the complexity and time demands.

6.2 CONTROLLABILITY MATRIX

6.2.1 Overview

Similar to the problem associated with system observability, as explained in section 4.3, controllability is another crucial property of a given system: **is it possible to maneuver the system in all possible configuration space, using certain manipulating inputs?**

The controllability of a system allows engineers to determine which states govern system behavior and dynamics, assess the feasibility of stabilization of unstable states, and settle on the most optimal controller design.

6.2.2 The Matrix

Controllability and observability are dual concepts that concern themselves with the same problem, and their formal derivations are analogous. The controllability matrix uses the state matrix and the input matrix (essentially, the dynamics equation) to assess the stability of a system with n state variables [22]:

$$\mathcal{C} = \begin{bmatrix} B & AB & \dots & A^{n-1}B \end{bmatrix} \tag{6.1}$$

If matrix $\mathcal{C}$ maintains *full rank*, then the system is controllable. In this calculation, the original, unaugmented A and B matrices were used, since the main goal of the controllability study is to **find out whether it is possible to use the forces as inputs to the proposed control scheme**. The unaugmented dynamics equation deploys the forces as inputs, and is therefore more appropriate to use.

In MATLAB, the `ctrb` function that calculates the controllability of a given system does not accept symbolic input arguments [31]. Therefore, a new function was written, found in appendix A.5 to calculate the controllability of the system.

1 A fuzzy-based controller is based on what is known as *fuzzy logic*, which computes a "degree of truth", rather than the conventional digital logic which relies on discrete 0's and 1's [30]

6.2.3 Results

The controllability matrix $\mathcal{C}$ does indeed maintain its full rank. This means that all states can be manipulated as desired.

6.3 MODEL PREDICTIVE CONTROL STRATEGY

6.3.1 Overview

Model Predictive Control, as described in section 2.4, is a control algorithm that is best suited for a model or a plant that comes with a certain set of constraints. This is because MPC is an optimal control sequence that can handle large time-delays, non-minimum phase plants, or unstable plants, while satisfying indispensable constraints, like safety constraints.

Since the main research goal is to be able to improve the control scheme such it is not vulnerable to external conditions, MPC presents a better control algorithm in comparison to its other alternatives, like PID or LQR controllers. While PID can be easily tuned, its perfomance is often only sufficient for SISO systems. LQR controllers can be implemented on MIMO systems, yet signal constraints are not very well addressed[2]. MPC, on the other hand, handles structural changes, such as sensor and actuator failure, wind, propeller or motor damage, and other factors that change system parameters and system structure.

6.3.2 Mathematical Formulation

The mathematical formulation of MPC relies on 3 main elements [33]:

- **Prediction Model:** MPC uses the system model to predict future outputs based on a pre-defined prediction model over a pre-defined prediction horizon. The prediction horizon is defined as $N_y = N_2 - N_1$, where N_1 and N_2 are the lower and upper horizons respectively. This represents the number of future control intervals that the MPC must evaluate by prediction [33][34].

- **Cost Function:** the cost function is often formulated to ensure that the output y tracks the reference trajectory over the pre-defined upper prediction horizon N_2. fig. 6.1 depicts this explanation quite well. A cost function may make use of one or multiple weighting matrices such that the output target trajectory is reached by repeating the predcition and optimization in each time instant [33] [35].

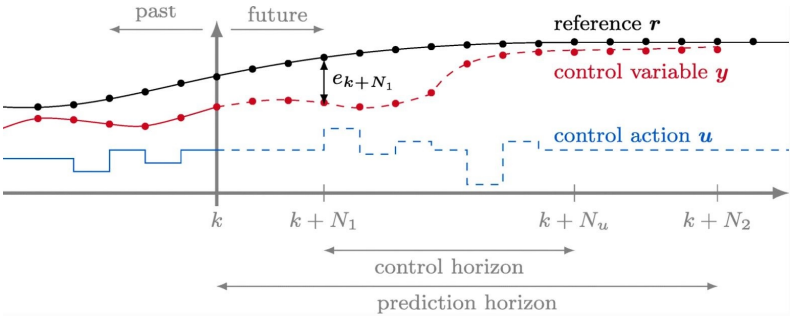


Figure 6.1: The principle of the cost function of model predictive control. From [35]

2 LQR computes and uses only one optimal solution for the entire time horizon, whereas MPC computes a new optimal solutions in regular sample intervals, thus addressing said constraints better [32]

- **Control Laws:** defined by the control horizon and its associated parameters, the control horizon is the number of moves made by the manipulated variables such that they are optimized at a control interval.

MPC builds upon classical control theory, as can be concluded from the explanation above. In essence, MPC directly and explicitly considers the system constraints (defined by the states, inputs, outputs, control and prediction horizons, and the cost function) to optimize a control sequence that minimizes said cost function [34][35]. This process is visualized in fig. 6.2. Because of MPC's ability to predict and optimize at every time instant, the control sequence solution is optimal for the defined finite time horizon [36].

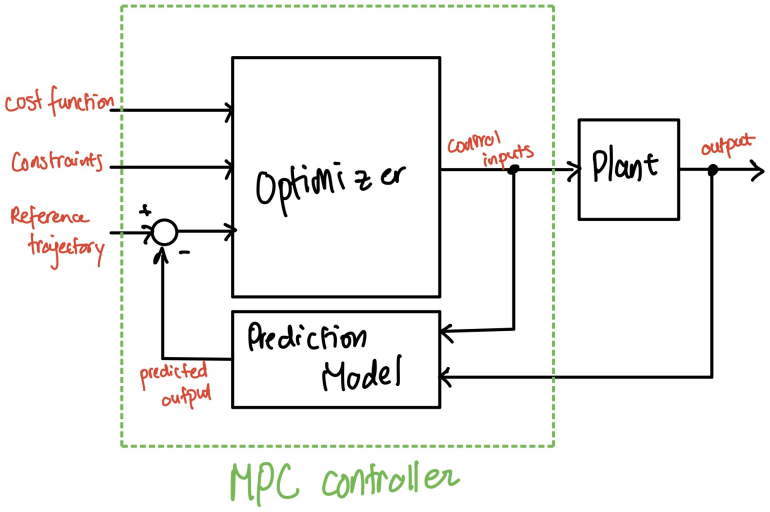


Figure 6.2: Block diagram showing the overall process of how MPC controllers optimize the control sequence [34]

6.3.3 MPC for Flight Control

The design parameters required to implement MPC are plenty, as explained previously, and the process of choosing and tuning these parameters is arduous and nontrivial. As P. Miotto and R. C. LePome stated at the AIAA GUidance, Navigation, and Control Conference and Exhibit, controller design becomes more of an art than a science when dealing with MPC [37].

Finding out which control parameters to use to control the attitude and position of the birotor model requires the mathematical formulation to be figured out in great detail, lots of tuning, and a lot of time dedicated. Due to the time constraints of this thesis assignment, the modeling of MPC is outside the scope of this research.

MATLAB offers a Model Predictive Control toolbox that allows engineers to specifically design model-predictive controllers. In concatenation with MATLAB's Aerospace Blockset, the UAV can be simulated with high fidelity, to simulate, analyze, and optimize system behavior.

6.4 CONCLUSION

In this chapter, the linearized model was found to be controllable. This allows for the implementation of a controller. The idea and the possibility of implementing MPC were explored. The appropriate control inputs and algorithm were discussed, and a clear plan for the implementation steps was laid out. However, due to the

limited time dedicated to this research and the nuanced complexity of MPC, the controller was not implemented in Simulink nor tested. However, this thesis may still serve as solid grounds for continuing the research on this topic and implementing an MPC.

7 DISCUSSION, CONCLUSION & RECOMMENDATIONS

7.1 INTRODUCTION

In the introduction of this thesis report, section 1.3, several research questions and goals have been elaborated, with an intention to find answers and accomplish said goals throughout the phases of this research. In this chapter, a summary will be provided: the most important discussion points will be addressed, some recommendations will be proposed, and the research answers and most important findings will be concluded.

7.2 DISCUSSION

7.2.1 Can the system be observed, taking into consideration the added flexibility to the model, as well as the forces?

The flexibility introduced to the model manifested itself in elastic potential terms, which added complexity to the non-linear model and made it hard to work with. Based on various assumptions, the model was linearized and put into state space representation. The observability of this linearized system was then studied, after augmenting the states to include the thrust forces as well. The observability matrix retained its full rank, meaning that the system is fully observable. However, the observability study did exhibit some unusual results. By making such strong assumptions about the non-linear model, the observability study conveyed that the system depends on the original model assumptions. However, since this research is focused on one particular model and has purely theoretical goals, this model reliance was not an issue for the continuation of this research.

7.2.2 Is a state observer feasible to implement?

This research set out to estimate the forces in two ways: an open loop estimator, and a closed loop state observer. The open loop estimator merely used the linearized state space equations to estimate the forces from the readily available sensor measurements. The results showed that, given the linear model still closely resembles the non-linear model at hand, the forces can be estimated fairly well, with a mean absolute error of $0.017N$. The results also showed that the estimator is sensitive to noise: the error increases when there are any jitters in the inputs. In an attempt to solve this, a closed loop state observer was implemented. No results were yielded due to systematic and computational simulation errors, and lack of time to debug such errors.

7.2.3 Is the system controllable?

By conducting a controllability study, the linear system dynamics are fully controllable. This means that all definite system states can be reached from any fixed initial state fed into a controller in a finite amount of time. This result is considered an important property of the control system to be defined. The design of the controller

can then start by making a simple PID controller for a MIMO system, since PID controllers are easier to tune, before trying to implement a more complex algorithm such as MPC. In this way, more concrete, comparative conclusions can be drawn.

7.3 RECOMMENDATIONS

7.3.1 The Optimal Kalman Filter

In control theory, the Optimal Kalman Filter is an optimal estimation algorithm that is typically used when a dynamic system has states that are subject to stochastic processes (i.e. Gaussian processes): the state cannot be measured directly from the available sensor data or the sensor data is subject to noise. A Kalman Filter estimates such states by using a series of measurements observed over time, and by estimating the *joint probability distribution*, produces an estimate of the unknown variable in question. The intricate workings of a Kalman Filter were discussed in section 2.3.

The complexity of a Kalman Filter is justified when an optimal estimation algorithm is needed. If, after the implementation of the Luenberger Observer, the results showed that estimating the states is subject to noise, a Kalman Filter may then be implemented.

An extended Kalman Filter might even be used for the non-linear birotor model.

7.3.2 Fully-Actuated Model

In partial fulfillment of his bachelor's degree, fellow student Idse Kuijper investigated the possibility of estimating the forces more accurately. Essentially, Kuijper compared two dynamical models: one with 3 differently oriented propellers with 3 strain gauges mounted on each arm, in contrast to a system with 3 propellers but with only 2 strain gauges. The author found that the lack of a strain gauge in the latter model manifested in a deficiency in observability, whereas the former system with 3 sensors was fully observable. This is because the 2 strain gauges only allow for the small angle deflections and their derivatives to be measured, producing a linear combination of all 3 forces of the propellers, whereas the 3 strain gauges allowed the all 3 forces to be measured independently. the author also implemented a generalized 3D controller, which showed decent performance under the condition that the tilt angle ϕ was zero [38]. This means that to sufficiently control the rotation and the attitude of the UAV, a more complex control scheme, such as the controller design discussed in this thesis, presents a better solution to this control problem.

7.4 CONCLUSION

This research provided an analysis on the feasibility of using elastic strain gauges that measure small angle deflection on multirotor arms to estimate the forces empirically, and use these forces as control inputs. Throughout this research, some interesting answers were found: the linear system is fully observable and controllable. As far as the goals of the research go, the observability of the system was studied, the dynamic system was successfully decoupled to augment and estimate the states of the system, including the propeller forces. Unfortunately, the control-related goals were not accomplished. Nevertheless, some controller designs were explored and elaborated, thus laying down a strong foundation for further research and progress to come. In conclusion, the topics learned throughout the bachelor's degree were revisited and utilized to a greater depth and new control theories were explored.

BIBLIOGRAPHY

[1] E. Alvarado, *237 Ways Drone Applications Revolutionize Business — Droneii*, en-US, May 2021. [Online]. Available: https://droneii.com/237-ways-drone-applications-revolutionize-business (visited on 06/30/2022).

[2] E. Pakken, "Design, modeling, and control of a novel multirotor UAV with embedded 3D-printed thrust force sensor — Robotics and Mechatronics," 2022. [Online]. Available: https://www.ram.eemcs.utwente.nl/education/assignments/design-modeling-and-control-novel-multirotor-uav-embedded-3d-printed-thrust.

[3] J.-J. Xiong and E.-H. Zheng, "Optimal Kalman Filter for state estimation of a quadrotor UAV," en, *Optik*, vol. 126, no. 21, pp. 2862–2868, Nov. 2015, ISSN: 0030-4026. DOI: 10.1016/j.ijleo.2015.07.032.

[4] *State Observers and Kalman Filters*, en.

[5] R. Kálmán, "A new approach to linear filtering and prediction problems" transaction of the asme˜journal of basic," 1960. DOI: 10.1115/1.3662552.

[6] J. A. H. Naranjo, H. Murcia, C. Copot, and R. Keyser, "Model predictive path-following control of an A.R. drone quadrotor," en, 2014.

[7] M. Nikolaou, "Model predictive controllers: A critical synthesis of theory and industrial needs," 2001. DOI: 10.1016/S0065-2377(01)26003-7.

[8] *Mathematical model — Britannica*, en. [Online]. Available: https://www.britannica.com/science/mathematical-model (visited on 06/30/2022).

[9] J. J. Craig, *Introduction to robotics: mechanics and control*, eng, 3. ed., international ed, ser. Pearson education international. Upper Saddle River, NJ: Pearson, Prentice Hall, 2005, ISBN: 9780131236295 9780201543612.

[10] L. N. Hand and J. D. Finch, *Analytical mechanics*. Cambridge ; New York: Cambridge University Press, 1998, ISBN: 9780521573276 9780521575720.

[11] A. De Luca, *Dynamic Model of Robots: Lagrangian Approach*, English, Robotics 2 Lecture, Sapienza University of Rome. [Online]. Available: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/http://www.diag.uniroma1.it/deluca/rob2_en/03_LagrangianDynamics_1.pdf.

[12] L. D. Hoffmann, G. L. Bradley, and K. H. Rosen, *Calculus for business, economics, and the social and life sciences*, 8th ed. Boston: McGraw Hill Higher Education, 2004, ISBN: 9780072424324.

[13] R. Feynman, *The Feynman Lectures on Physics Vol. II Ch. 19: The Principle of Least Action*. [Online]. Available: https://www.feynmanlectures.caltech.edu/II_19.html (visited on 06/30/2022).

[14] H. P. Gavin, *Matrix Structural Analysis*, English, CEE 421L, Duke University, 2012.

[15] L. A.B. Torres, *Robot Dynamic - Part I: Euler-Lagrangian Approach*, English, Lecture, Federal University of Minas Gerais. [Online]. Available: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/http://www.cpdee.ufmg.br/~torres/wp-content/uploads/2018/02/Robot_Dynamics_part_1.pdf.

[16] *Symbolic Math Toolbox*, en. [Online]. Available: https://nl.mathworks.com/products/symbolic.html (visited on 06/30/2022).

[17] A. Franchi, *Control Engineering*, English, Version-2021.3, ser. Systems and Control for EE. Enschede, Netherlands: Robotics and Mechatronics, University of Twente, Nov. 2021.

[18] T. Hägglund, *State Space Models, Linearization, Transfer Function*, Automatic Control, Basic Course, Lecture 2, Lund University, Oct. 2019.

[19] E. W. Weisstein, *Jacobian*, en, Text. [Online]. Available: https://mathworld.wolfram.com/ (visited on 06/30/2022).

[20] *Jacobian matrix - MATLAB jacobian - MathWorks Benelux*. [Online]. Available: https://nl.mathworks.com/help/symbolic/sym.jacobian.html (visited on 06/30/2022).

[21] R. E. Kalman, "On the general theory of control systems," en, *IFAC Proceedings Volumes*, 1st International IFAC Congress on Automatic and Remote Control, Moscow, USSR, 1960, vol. 1, no. 1, pp. 491–502, Aug. 1960, ISSN: 1474-6670. DOI: 10.1016/S1474-6670(17)70094-8. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474667017700948 (visited on 06/30/2022).

[22] R. W. Zhang, *Observability and Controllability*, English. [Online]. Available: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.ece.rutgers.edu/~gajic/psfiles/chap5traCO.pdf.

[23] *Observability matrix - MATLAB obsv - MathWorks Benelux*. [Online]. Available: https://nl.mathworks.com/help/control/ref/ss.obsv.html (visited on 06/30/2022).

[24] S. Boyd, *Observability and state estimation*, EE263, University of Stanford, 2007.

[25] *Null space of matrix - MATLAB null - MathWorks Benelux*. [Online]. Available: https://nl.mathworks.com/help/matlab/ref/null.html (visited on 06/30/2022).

[26] Z. Gajic, *Introduction to Linear and Nonlinear Observers*, Lecture, Rutgers University.

[27] J. H. Davis, "Luenberger Observers," en, in *Foundations of Deterministic and Stochastic Control*, ser. Systems & Control: Foundations & Applications, J. H. Davis, Ed., Boston, MA: Birkhäuser, 2002, pp. 245–254, ISBN: 9781461200710. DOI: 10.1007/978-1-4612-0071-0_8. [Online]. Available: https://doi.org/10.1007/978-1-4612-0071-0_8 (visited on 06/30/2022).

[28] *Understanding Kalman Filters, Part 2: State Observers Video*, en. [Online]. Available: https://nl.mathworks.com/videos/understanding-kalman-filters-part-2-state-observers-1487694734527.html (visited on 06/30/2022).

[29] G. Ellis, *Control system design guide: using your computer to understand and diagnose feedback controllers*, Fourth edition. Amsterdam: Elsevier/BH, 2012, ISBN: 9780123859204.

[30] V. Novák, I. Perfilieva, and J. Močkoř, *Mathematical principles of fuzzy logic*, ser. Kluwer international series in engineering and computer science secs 517. Boston: Kluwer Academic, 1999, ISBN: 9780792385950.

[31] *Controllability matrix - MATLAB ctrb - MathWorks Benelux*. [Online]. Available: https://nl.mathworks.com/help/control/ref/ss.ctrb.html (visited on 06/30/2022).

[32] L. Wang, *Model predictive control system design and implementation using MATLAB*, ser. Advances in industrial control. London: Springer, 2009, OCLC: ocn403385816, ISBN: 9781848823303 9781848823310.

[33] Z.-j. Yang, X.-h. Qi, and G.-l. Shan, "Simulation of flight control laws design using model predictive controllers," in *2009 International Conference on Mechatronics and Automation*, ISSN: 2152-744X, Aug. 2009, pp. 4213–4218. DOI: 10.1109/ICMA.2009.5246511.

[34] *Understanding Model Predictive Control*, en. [Online]. Available: https://nl.mathworks.com/videos/series/understanding-model-predictive-control.html (visited on 07/01/2022).

[35] M. Schwenzer, M. Ay, T. Bergs, and D. Abel, "Review on model predictive control: An engineering perspective," en, *The International Journal of Advanced Manufacturing Technology*, vol. 117, no. 5, pp. 1327–1349, Nov. 2021, ISSN: 1433-3015. DOI: 10.1007/s00170-021-07682-3. [Online]. Available: https://doi.org/10.1007/s00170-021-07682-3 (visited on 07/01/2022).

[36] L. Singh, "Autonomous Missile Avoidance Using Nonlinear Model Predictive Control," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, ser. Guidance, Navigation, and Control and Co-located Conferences, American Institute of Aeronautics and Astronautics, Aug. 2004. DOI: 10.2514/6.2004-4910. [Online]. Available: https://arc.aiaa.org/doi/10.2514/6.2004-4910 (visited on 07/01/2022).

[37] P. Miotto and R. LePome, "Design of a Model Predictive Control Flight Control System for a Reusable Launch Vehicle," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, ser. Guidance, Navigation, and Control and Co-located Conferences, American Institute of Aeronautics and Astronautics, Aug. 2003. DOI: 10.2514/6.2003-5360. [Online]. Available: https://arc.aiaa.org/doi/10.2514/6.2003-5360 (visited on 07/01/2022).

[38] I. Kuijper, "Fully-actuated multirotors embedding elastic sensors to measure arm deflection: System modeling and control," 2022.

A MATLAB EXCERPT

A.1 JACOBIAN LINEARIZATION

```
%% jacobian linearization
%define A and B matrices
A = jacobian(dx_state,x_state);
A = subs(A, x_state, x0);
A = subs(A, u, u0);
A = simplify(A);

B = jacobian(dx_state,u);
B = subs(B, x_state, x0);
B = subs(B, u, u0);
B = simplify(B);

%state space representation
dx_state_linear = A*x_state + B*u;
```

A.2 OBSERVABILITY CALCULATION

```
%% observability
%since obsv() matlab function does not accept symbolic input arguments, the
%calculation of observability has to be done manually

%define the dimensions of the observability matrix
n = size(tilde_A,1);
ny = size(tilde_C,1);

%allocate ob symbolically and compute each C A^k term
ob = sym('ob',[n*ny n]);
ob = subs(ob, ob(1:ny,:), tilde_C);

for k=1:n-1
  ob = subs(ob,ob(k*ny+1:(k+1)*ny,:),ob((k-1)*ny+1:k*ny,:) * tilde_A);
end

ob = simplify(ob);

%the number of unobservable states is the number of state variables - rank
%of the observability matrix
unob = length(tilde_A) - rank(ob);
```

A.3 LINEAR MODEL INVERSION

```
function [f1, f2] = inversion(db, ddb, UAV)
% INPUTS
% db        :     vector including dbs and dbd
% ddbd      :     vector including ddbs and ddbd
% UAV       :     calls parameters.m to define UAV specifications
% OUTPUTS
% [f1, f2]  :     vector including the two forces to be estimated

% UAV specifications to be called
```

```
J = UAV.J(2,2);
k = UAV.K;
L = UAV.L_tot;
m1 = UAV.mass_arm; % mass of the propeller
m = UAV.mass; % mass of the UAV
g = 9.81;

% separate the inputs
dbs = db(1);
dbd = db(2);
ddbs = ddb(1);
ddbd = ddb(2);

% define matrix B
B = [-1/(L*m1), 1/(L*m1);...
     -1/(L*m1), -1/(L*m1)];

% define the two entries from matrix A
a1 = -(k*(m + 2*m1))/(L^2*m*m1);
a2 = -(k*(2*m1*L^2 + J))/(J*L^2*m1);

% linear model inversion
y = B\[ddbs - a1*dbs; ddbd - a2*dbd];

% adding the operating point
f1 = y(1) + 22.446;
f2 = y(2) + 22.446;

end
```

A.4 TRAJECTORY PLANNING

```
%% initial condtioins

% pos
init.p = [0;0;0];
% velocity (world-frame)
init.v = [0;0;0];
% orientation (Rotation Matrix)
init.R = Rz(0)*Ry(0)*Rx(0);
% angular velocity (body-frame)
init.w = [0;0;0];

init.B_dot = [0;0];
init.B = ...
    [-0.0018664783741618794096049504710784;0.0018664783741618794096049504710784];
%-0.00000252259684796950008583123763500061 for K = 6,2..E4;
% orientation (Quaternion)
init.q = rotm2quat(init.R);
% tilting angles
init.u_v = deg2rad(0);

% important: compute nominal rotor velocities to sustain gravity given
% initial orientation and initial tilting angles
init.u_lambda = [0;0;(UAV.mass+2*UAV.mass_arm)*env.g];

%% trajectory parameters

% position
traj.p_start = init.p;
traj.p_end = init.p; % or [0.5;0;0.5]; or [1;0;1]; for different test ...
    conditions

% orientation
traj.R_start = init.R;
traj.R_end = Rz(0)*Ry(0)*Rx(0);

% trajectory time
```

```
traj.T = 10;

%% elementary rotations
function R = Rx(a)
R  = [1     0       0 ;
      0 cos(a) -sin(a);
      0 sin(a)  cos(a)];
end

function R = Ry(a)
R  = [ cos(a) 0 sin(a);
           0  1     0 ;
      -sin(a) 0 cos(a)];
end

function R = Rz(a)
R  = [cos(a) -sin(a) 0;
      sin(a)  cos(a) 0;
          0       0  1];
end
```

A.5 CONTROLLABILITY CALCULATION

```
n = size(A,1);
nu = size(B,2);

co = sym('co',[n n*nu]);
co = subs(co,co(:,1:nu),B);
for i=1:n-1
  co = subs(co,co(:,i*nu+1:(i+1)*nu), A * co(:,(i-1)*nu+1:i*nu));
end

unco = length(A) - rank(co);
```

A.6 MEAN ABSOLUTE ERROR

```
% This function calculates the mae of a signal with reference to original ...
    signal
% OUTPUT
% mae     :     mean absolute error
%
% INPUTS
% orgSig  :     original (reference) signal
% recSig  :     reconstructed (estimated) signal


function mae = fcn(orgSig,recSig)

    absErr = norm(orgSig(:)-recSig(:),1);
    mae = absErr/length(orgSig(:));

end
```

B SIMULINK

B.1 MODEL OVERVIEW

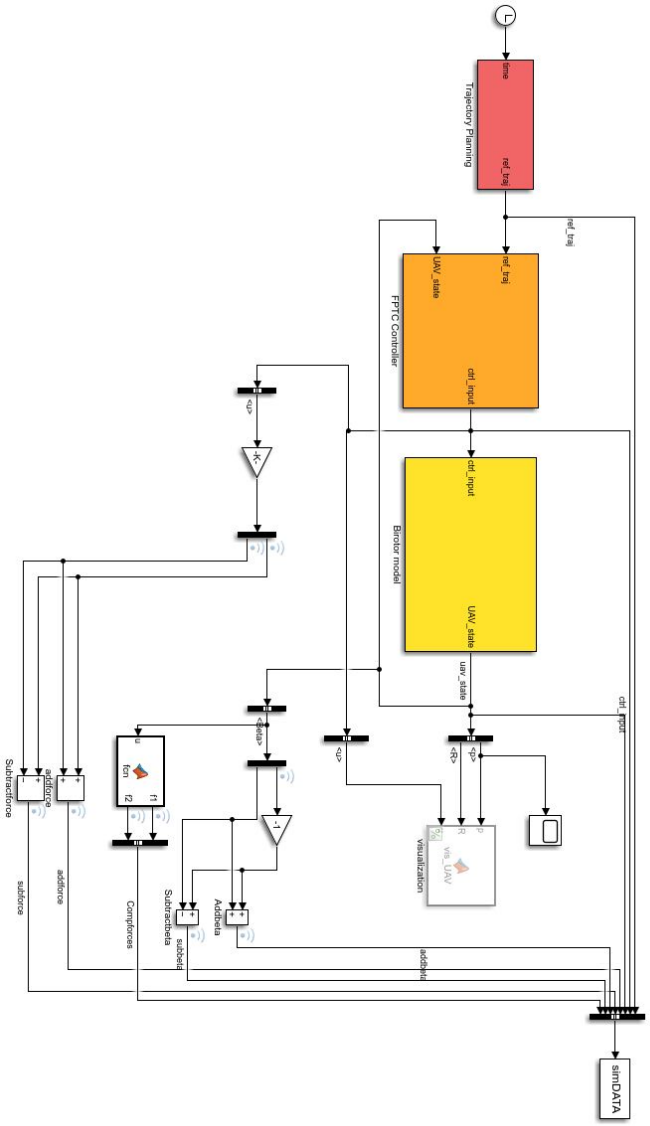
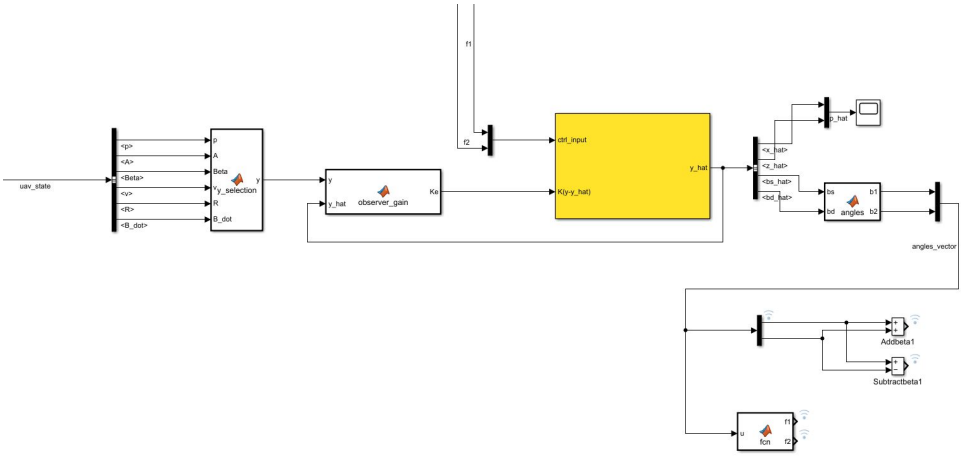
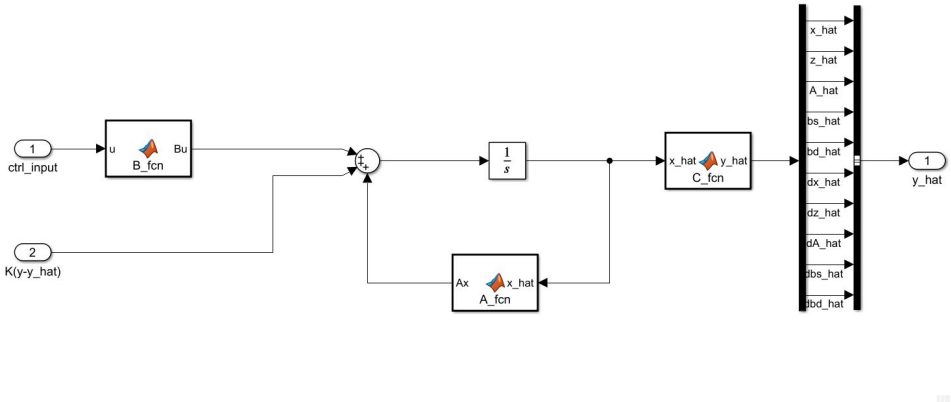


Figure B.1: Overview of the birotor model in Simulink (rotates 90°)

B.2 LUENBERGER OBSERVER



(a) Overview of the state observer, showing the selection of the UAV states, the observer gain block that multiplies $y - \hat{y}$ by observer gain K, as well as the feedback loop.

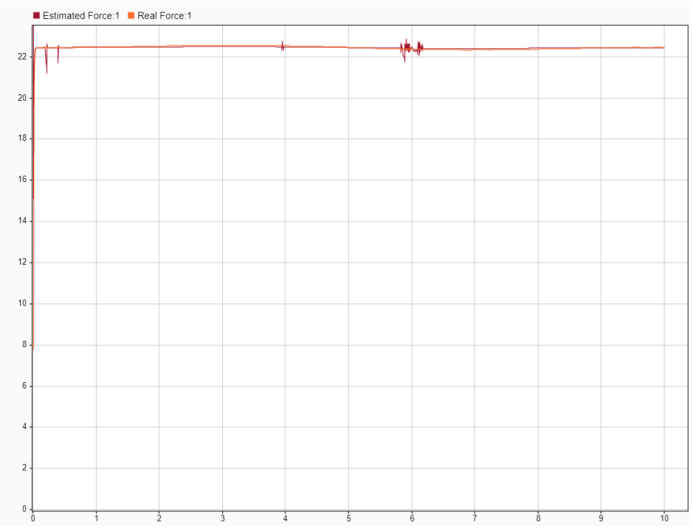


(b) The Observer block, showing how the states are reconstructed using the state-space model.

Figure B.2: State observer implementation in Simulink.

C SIMULATION RESULTS

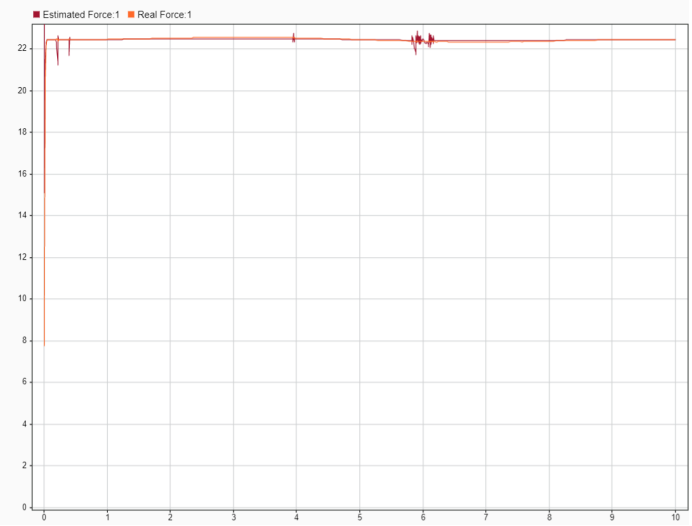
C.1 FORCE–TIME GRAPHS



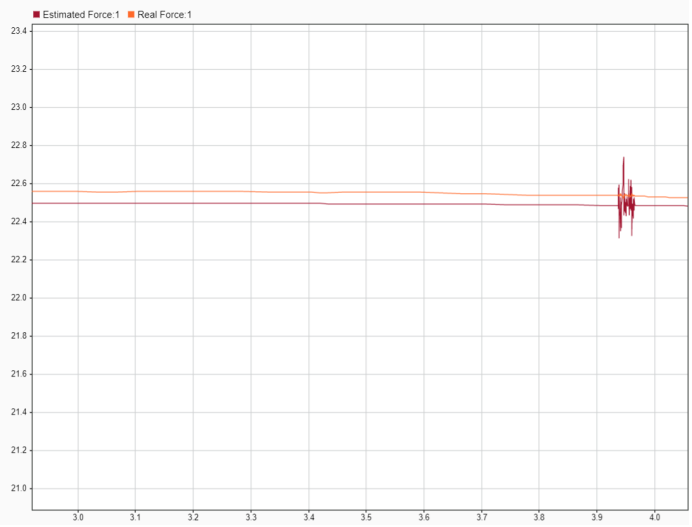
(a) Full-scale.

(b) Close-up.

Figure C.1: The force (N) - time (s) graph of the estimated forces and the measured forces, given a gentle moving trajectory.



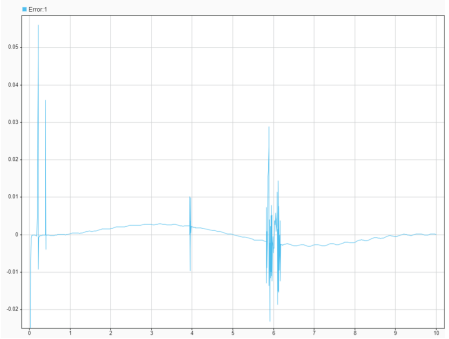
(a) Full-scale.



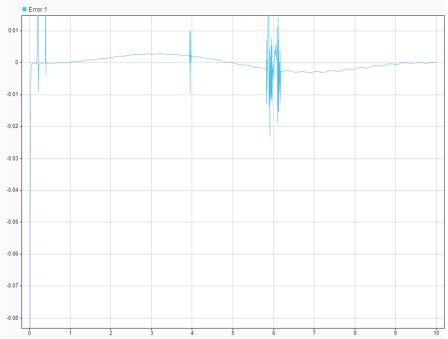
(b) Close-up.

Figure C.2: The force (N) - time (s) graph of the estimated forces and the measured forces, given an aggressive moving trajectory.

C.2 ERROR-TIME GRAPHS



(a) Error-time plot for the second test: the UAV was moving gently from $p = (0, 0, 0)$ to $p = (0.5, 0, 0.5)$.



(b) Error-time plot for the second test: the UAV was moving more aggressively from $p = (0, 0, 0)$ to $p = (1, 0, 1)$.

Figure C.3: Error (N) - time (s) graphs.

COLOPHON

This document was typeset using LaTeX. The document layout was generated using the arsclassica package by Lorenzo Pantieri, which is an adaption of the original classicthesis package from André Miede.

ROBOTICS
AND
MECHATRONICS

UNIVERSITY
OF TWENTE.