Fast Nonlinear Model Predictive Control of Quadrotors: Design and Experiments

by

Hadi Mohammadi Daniali

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
System Design Engineering

Waterloo, Ontario, Canada, 2019

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Quadrotor (or quadcopter) is a type of Unmanned Aerial Vehicles (UAVs). Due to the quadrotors simple and inexpensive design, they have become popular platforms. This thesis proposes a computationally fast scheme for implementing Nonlinear Model Predictive Control (NMPC) as a high-level controller to solve the path following problem for unmanned quadrotors. After discussing the background and reviewing the literature, it is noted that this problem referred widely in the literature as a necessary step toward the autonomous flight of quadrotor UAVs. The previous studies usually used simplified models which are computationally uncomplicated and straightforward in terms of control developments and stability investigations. Moreover, some articles are presented showing the importance of accurate state observation on the performance of feedback-based control approaches.

The NMPC-based controller is designed using a more realistic highly nonlinear control-oriented model which requires heavy computations for practical real-time implementations. To deal with this issue, the Newton generalized minimal residual (Newton/GMRES) method is applied to solve the NMPC's real-time optimizations rapidly during the control process. This technique uses the Hamiltonian method to derive a set of equations with multiple variables. To solve these in a real-time application, the Newton/GMRES method applies forward-difference generalized minimal residual (fdgmres) algorithm.

The simulation and experimental result using a commercial drone, called AR.Drone 2.0, in our laboratory instrumented by a Vicon Vantage motion capture system, demonstrate that our feedback-based control method's performance highly depends on the reliability of the state vector feedback signals. As a result, a Kalman filter and Luenberger observer algorithms are used for estimating unknown states. The NMPC-based controller operation is simulated, and the result reveals the similar efficiency of observers. Moreover, the NMPC control approach is compared with a proportional controller which shows great improvements in the response of the quadrotor. The experiment showed that our control method is sufficiently fast for practical implementations, and it can solve the trajectory tracking problem properly even for complex paths. This thesis is concluded by stating a summary of contributions and some potential future works.

Acknowledgements

I would like to thank my supervisor, Prof. Azad who made the completion of this thesis possible.

I acknowledge the Natural Sciences and Engineering Research Council of Canada for supporting this study.

Finally, I would like to thank my family and all of my friends for all their support.

Dedication

This is dedicated to the one I love.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview and Motivation

Quadrotor (or quadcopter) is a type of Unmanned Aerial Vehicle (UAV) consists of a rotating rigid structure with six degrees of freedom, and four brush-free motors. All quadcopters have two identical pairs of rotors. One pair rotates clockwise, while the other rotates counter-clockwise. This platform is an ideal device to explore unknown and out of reach environments. Because of their simple structure, quadrotors are usually cheaper compared to other types of rotary and fixed-wing UAVs. Due to the capabilities of quadrotors to hover, vertical take-off, and landing, they have become popular platforms for civilian and military duties; such as photography, security, search and rescue operations, drone-delivery, agricultural activities, art, sport, journalism, etc. [1, 18, 28, 38]. To accomplish these tasks properly, quadrotors should be able to fly autonomously. To do so, they should have the capability to fly in plan paths, and track trajectories [63, 67].

Path following is one of the most significant problems discussed in the literature in recent years and a wide variety of control algorithms have been applied to solve this problem for quadrotor UAVs. The control approaches to solve this problem can be classified as low-level and high-level controllers. The low-level controller receives the direction of the desired movements from the user and computes the desired rotors input. By applying the rotors input, the controller can move the drone toward the desired direction. On the other hand, by knowing the desired position and orientation or the path of the quadrotor, the high-level controller defines the desired motion.

The performance of any feedback-based controller highly depends on the accuracy of the feedback signal which represents the state of the system. Therefore, sensors and filters

are employed to determine the state of the system. Commercial quadrotors usually have some onboard sensors, such as global positioning system (GPS), inertial measurement unit (IMU), radio detection and ranging (radar), sound navigation ranging (Sonar), light detection and ranging (Lidar), camera, etc. However, these types of sensors usually do not provide a reliable feedback signal. Therefore, in this study, parallel to onboard sensors, we use a more precise off-board sensor to alleviate the foregoing drawback.

1.2 Objectives

The objective of this thesis is to solve the trajectory tracking problem. In order to solve this problem, this thesis provides the following steps:

- Control approach:
  - In order to design a high-level controller, the system model that is a set of nonlinear equations between the input and output of the controller is obtained.
  - Nonlinear Model Predictive (NMPC) technique which is an optimal control method is employed to solve the trajectory tracking problem. the NMPC controller uses the obtained model as a control-oriented model.
  - Due to the computational cost of NMPC real-time optimization process, which also depends on the complexity of the control-oriented model inside the controller, the use of fast optimizers is critical to implement this approach in practice. Therefore, To tackle the NMPC's real-time optimization problem, the Newton generalized minimal residual (Newton/GMRES) approach is used. By applying Forward-Difference generalized minimal residual (fdgmres) algorithm, this method offers a quick scheme to calculate the Hamiltonian function's solution which can handle the real-time optimization problem properly [35, 47, 60].
- State estimation:
  - The NMPC technique is a feedback-based controller which needs observation of all states.
  - To check whether the estimations methods are implementable, the system observability is discussed.
  - To estimate the unknown states of the system, Kalman filter [61] and Luenberger Observer [30] techniques are used.

- Validation of the designed NMPC and estimators:
  - some simulations and experiments have been conducted with a commercial quadrotor (AR.Drone 2.0) by employing the MATLAB/Simulink toolboxes introduced in [55, 60]. In the tests, a set of off-board cameras (Vicon Vantage motion capture system) and an onboard IMU have been used to track the quadcopter's states.
  - Some results are provided without any state estimation techniques to show the importance of accurate observation.
  - The Kalman filter and Luenberger obsrver are compared.
  - The performance of the proposed NMPC-based controller is compared with a proportional controller.

1.3 Organization of the Thesis

The next chapters of this thesis discusses the background, the literature review, and the contribution.

Chapter 2 explains the motion of quadrotors, the low-level, and the high-level controllers in details. Then, this chapter reviews the literature on solving the trajectory tracking control for quadrotor UAVs, focusing on NMPC.

In Chapter 3, upon discussing the problem statement, a more realistic high-level model of quadrotors is presented which is highly nonlinear. Then, the NMPC technique, which guarantees the optimum result, is discussed. Due to the problem complexity of the NMPC's optimization problem, Newton/GMRES and fdgmres algorithms are presented to solve this problem. Then, the NMPC is simulated in MATLAB/Simulink, and its inferior performance shows the importance of developing state estimation methods. Moreover, before representing the experiment results, a brief experiment setup is presented. This part introduces AR.Drone 2.0 and Vicon Vantage motion capture system.

In Chapter 4, to measure unknown vertical velocity, some state estimations techniques are introduced. Upon checking the observability of the system, a Kalman filter and a Luenberger observer are designed. Then, these methods are simulated and as well as compared to show the effectiveness of sate estimation approaches. Then, the NMPC performance is compared with the simulation of a proportional controller. Finally, this chapter is concluded after representing some experimental results.

Chapter 5 presents a summary of this thesis, concluding remarks, and some suggestions as future work to expand this study.

Chapter 2

Background and Literature Review

In this chapter, the low-level and the high-level control approaches and the motion of quadcopters are discussed. Then, a brief literature review on different control techniques for quadrotors, mostly focused on NMPC, is presented.

2.1 Two-level Control of Quadrotor

The low-level controller inputs of any quadrotor are left/right, forward/backward, up/down, and rotation around itself (namely z-axis). The task of this controller is to receive these inputs from the user (his/her joystick) and calculate the desired rotor inputs in such a way that the drone follows the desired path. As shown in Figure 2.1, in order to move the quadrotor to left or right, the low-level controller decides to rotate the drone around its x-axis, then the sum of four forces generated by the rotors forces it to fly to left/right. Also, the controller rotates the quadrotor around its y-axis when the user commands forward/backward. Therefore, left/right and forward/backward commands can be defined as ϕ and θ references, where ϕ is the roll angle and θ is the pitch angle. Moreover, up/down and rotation commands can be recognized as the reference $\dot{z}$ and $\dot{\psi}$, which are vertical velocity and yaw rate, respectively.

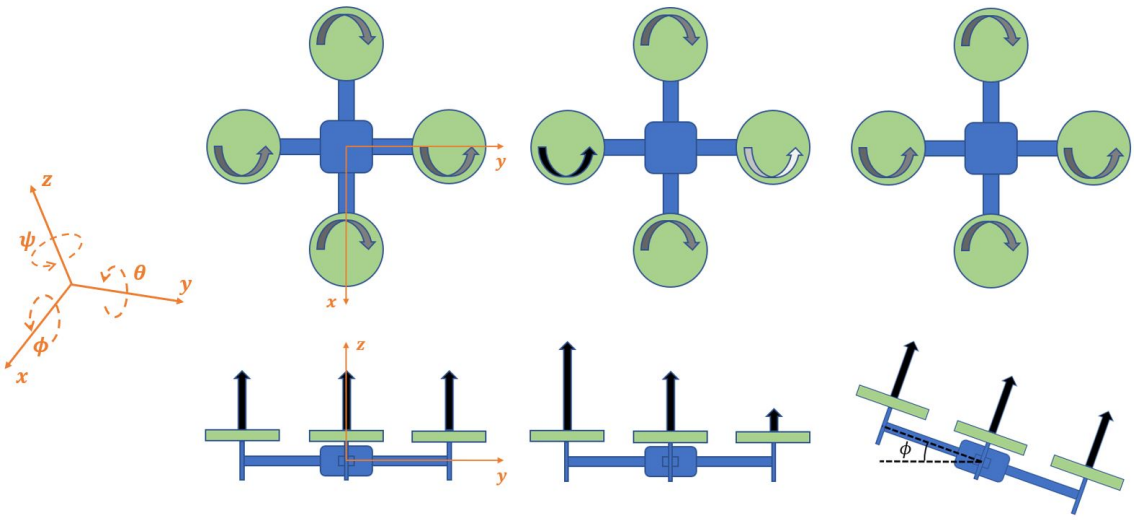

Figure 2.1: Quadrotor performing left/right movement

As depicted in Figure 2.2, to control the drone at a particular position or path, a high-level controller should obtain the desired position (x_d, y_d, and z_d) and heading (ψ_d), and generates $\dot{z}$, ϕ, θ, and $\dot{\psi}$ reference values to the low-level controller. Hence, the quadcopter and the low-level controller can be introduced as a plant which is going to be controlled by the high-level controller.

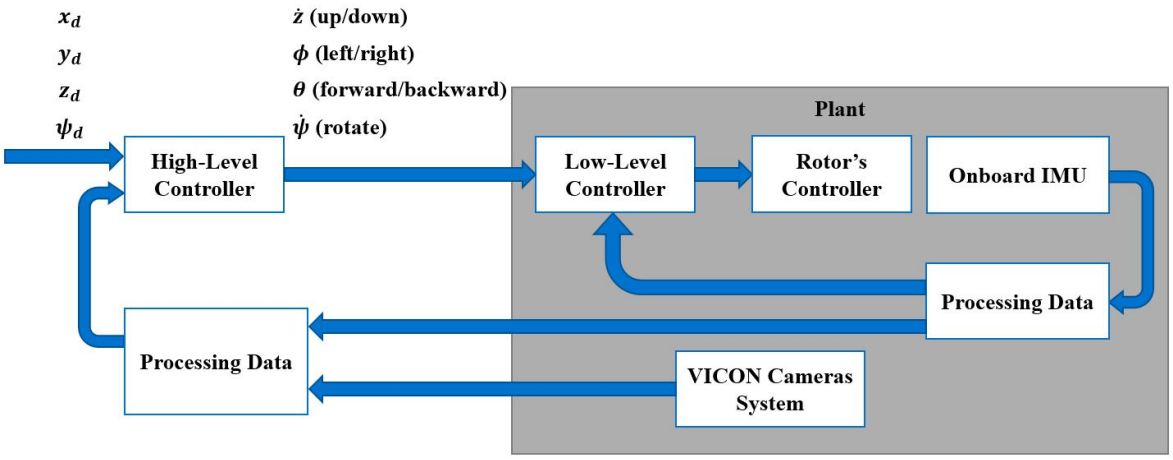

Figure 2.2: Two-level control architecture for drones

2.2 Literature on Quadrotor

This section, firstly, reviews the literature on control of quadrotor UAVs, then it discusses state estimation methods applied on quadrotors in previous studies.

2.2.1 Modeling and Control

Although the quadrotor's equations of motion are generally nonlinear, several types of linear control techniques have been investigated to find a solution to the path tracking problem. Due to the uncomplicated design and stability analysis of linear-based controllers, they are the most popular control methods. The reported results of linear controllers, such as Proportional-Integral-Derivative (PID) control, Linear Quadratic Regulator (LQR) control [13, 19, 36], and linear Model Predictive Control (MPC) [31], reveal that these kinds of control approaches are not capable of manipulating highly nonlinear systems like quadcopters appropriately since the linearized model ignores some dynamic terms.

In the literature, many techniques are applied to handle dynamics uncertainties. These uncertainties usually come from unknown parameters and simplified dynamics. Those approaches are widely applied to linear-based controllers which neglect some dynamics to

simplify the model. The most common method to overcome these issues is adaptive control which is classified as direct and indirect methods. The direct adaptive method calculates the control parameters directly by observing the input and the output of systems while the indirect approach computes the dynamic parameters, then the controller is designed based on the modified model.

Model Reference Adaptive Control (MRAC) is applied to quadrotor in [58]. Also, a direct MRAC method is employed in [24], in which a linear model with uncertain parameters is used. The method is compared with a nominal controller which reveals the superior performance of MRAC. Moreover, in [25], the MRAC and nominal approaches are compared with combined/composite model reference control (CMRAC). The CMRAC method uses a combination of direct and indirect adaptive techniques, and proposed tests show that adaptive approaches are very efficient for robust responses under uncertain conditions. [42] introduces Lyapunov-based MRAC method to handle instability caused by ignored uncertainties. The indirect adaptive technique is employed in [7] to design a linear parameter varying (LPV) controller. This study is not fully autonomous. Moreover, in [39] an indirect-based adaptive linear quadratic control (ALQC) is applied on a quadrotor. This investigation uses the least squares (LS) method for online parameter identification (PI).

Furthermore, the adaptive approach is applied to different types of neural network-based controller [73]. In [43], the backstepping method is combined with a radial basis function neural network (RBFNN) which is modified based on online learning. The simulation result verifies the effectiveness of the proposed method for quadrotors. The adaptive technique is also tried to be combined with nonlinear-based controllers. In [46], the Cerebellar Model Arithmetic Computer (CMAC), which is computationally fast and works well for direct adaptive technique, is used to approximate the nonlinear model. The study shows that this method cannot handle all types of uncertainties. [72] employs the robust integral of the signum of the error (RISE) and the immersion/invariance algorithms to design an adaptive-based nonlinear controller. This method can manage systems under unknown disturbances and uncertain parameters.

In [32, 33], a robust adaptive controller based on both linear and nonlinear models is applied on a quadrotor. This technique can handle different types of uncertainties, such as external forces and modelling errors by using Lyapunov-energy function. [17] proposes a Lyapunov-based backstepping approach to handle disturbances such as wind. To overcome the effects of this type of disturbances, the disturbance rejection method is employed. Also, the disturbance rejection method is applied in [65] as an answer to internal and external disturbances. This study utilizes a nonlinear feedback control based on backstepping method, combined with a robust disturbance-observer (DOB). Using H_∞ approach and modified

Rodrigues parameters (MRPs), the DOB method handles the involved uncertainty to find the attitude tracking error and follow the exact trajectory. Moreover, the DOB method is used in [64] to model uncertainties caused by the surrounding environment and manage them.

One of the most advanced robust model-based control techniques is Learning-Based Model Predictive Control (LBMPC) [15]. This method has been successfully applied to quadrotors in [8, 9]. Similar to any other type of MPC controllers, this technique predicts the state of the system and chooses the optimum input. Therefore, it is highly dependent on the accuracy of the control-oriented (prediction) model used inside the controller which represents the controlled system [18]. In order to get the best performance of a linear model at any instant, LBMPC should update the model's parameters online. However, it should be noted that linear models are not able to represent such a nonlinear platform accurately.

Also, the sliding mode control technique based on backstepping for quadrotors and also feedback-linearization technique discussed in [14] and [62], are different versions of nonlinear control methods. However, unlike the NMPC method, these approaches cannot impose directly constraints on the inputs, outputs, and states. Moreover, in [6] another feedback-linearization and backstepping approaches are presented which is not fully autonomous. Finally, in [20], the Lagrange technique is employed in order to find the quadrotor's model. However, the feedback signal is provided via cable.

2.2.2 State Estimation

State estimation plays a crucial role in the performance of feedback-based control methods. Therefore, in previous studies, many state estimation algorithms have been presented for quadrotor UAVs. These techniques establish methods to measure unknown sates by observing input/output. Also, these approaches are commonly used to diagnose faulty sensors. In [41], to overcome the effect of IMU's inaccurate data, an extended Kalman filter (EKF) and linear fixed-gain filter are designed and compared. Both filters provided more accurate attitude observations in contrast to typical methods. To accomplish this task, researchers used an improved dynamic model of quadrotor UAVs. Moreover, to consider the fault of sensors and actuators, [27] proposes a robust adaptive Kalman filter (RAKF). Based on the new data obtained from the quadrotor, the RAKF updates Kalman filter's parameters (process and measurement covariances). This method may provide precise estimation, even in the case of sensors or actuators failure. Also, [68] estimates the position and attitude of quadrotor by optimal Kalman filter (OKF) technique. In [66], the process

and measurement noises covariances are computed by autocovariance least-square (ALS) algorithm. Then, based on these parameters, an EKF method was developed, and its accuracy is tested by simulation.

Sensor fusion is another approach to filter inaccurate measurements which is applied in [2]. This study uses data obtained from an IMU, a sonar, and an optic-flow based vision system. This data fusion technique utilizes an EKF algorithm. Also, [52] develops and compare different strategies to estimate the state vector by fusing measurements obtained by IMU, GPS, and distance measurement sensors. Methods investigated in this article were EKF, unscented Kalman filter (UKF), particle filter (PF), and derivative-free nonlinear Kalman filter (DKF), and the result showed that the UKF and DKF algorithms are more efficient than other approaches. In [44], firstly, visual-inertial odometry (VIO) algorithm was developed to observe some of the states by stereo cameras. Then, a UKF-based data fusion method was designed to merge data obtained by VIO, IMU, and downward-pointing distance sensor.

Furthermore, [57] fuses data from multiple sensors which are measuring the state vector at different rates. Also, to overcome the effect of loss of data, this study developed an EKF observer. Moreover, for outdoor application, [56] designs a Kalman filter for data fusion obtained by GPS and AR.Drone Parrot quadrotor onboard sensors. Also, [40] employed Vicon motion capture as an off-board optical tracking system and an onboard IMU for small UAVs. Due to the tiny size of quadrotors, the Vicon data is noisy and unreliable. Hence, an EKF strategy developed in [11] is used to fuse data measured by these two sensors.

2.3 Literature on NMPC

Optimal control is one of the most popular control techniques vastly discussed in the literature. However, it was not an interesting approach to control quadrotors [38]. This approach chooses the optimal input to follow the best trajectory with the minimum energy consumption. This technique uses a cost function which consists of the input and the error at each step-time. LQR, H_∞, and MPC are different types of optimal control. In addition to the mentioned LQR control on quadrotors, adaptive LQR which updates the control-oriented model is presented and experimented in [16, 39]. Also, H_∞ is studied on quadrotor UAVs in [22] which employs a simplified nonlinear model to design a nonlinear H_∞ for a 2-DOF system. Moreover, [51] suggests a robust nonlinear H_∞ and integral MPC model. However, in contrast to MPC, LQR and H_∞ cannot consider equality and inequality constraints directly.

MPC is a closed-loop constrained optimal control. The block diagram of the MPC controller is shown in Figure 2.3. It consists of two parts, namely; Optimizer and control-oriented model. Control-oriented model is an internal dynamic model of the system which can predict the future output given the current state as a feedback signal and the future data. Also, the optimizer is capable of choosing the optimum input by solving the optimization problem. Moreover, this controller can supervise the Multi-Input Multi-Output (MIMO) control systems. In the MPC control approach, the optimizer minimizes the error between the predicted state calculated based on the control-oriented model and the desired trajectory, by choosing the most efficient control sequence. Then, it implements the first optimum control, and this process repeats at the next sampling time. Therefore, this approach is also introduced as Receding Horizon Control [5, 60].

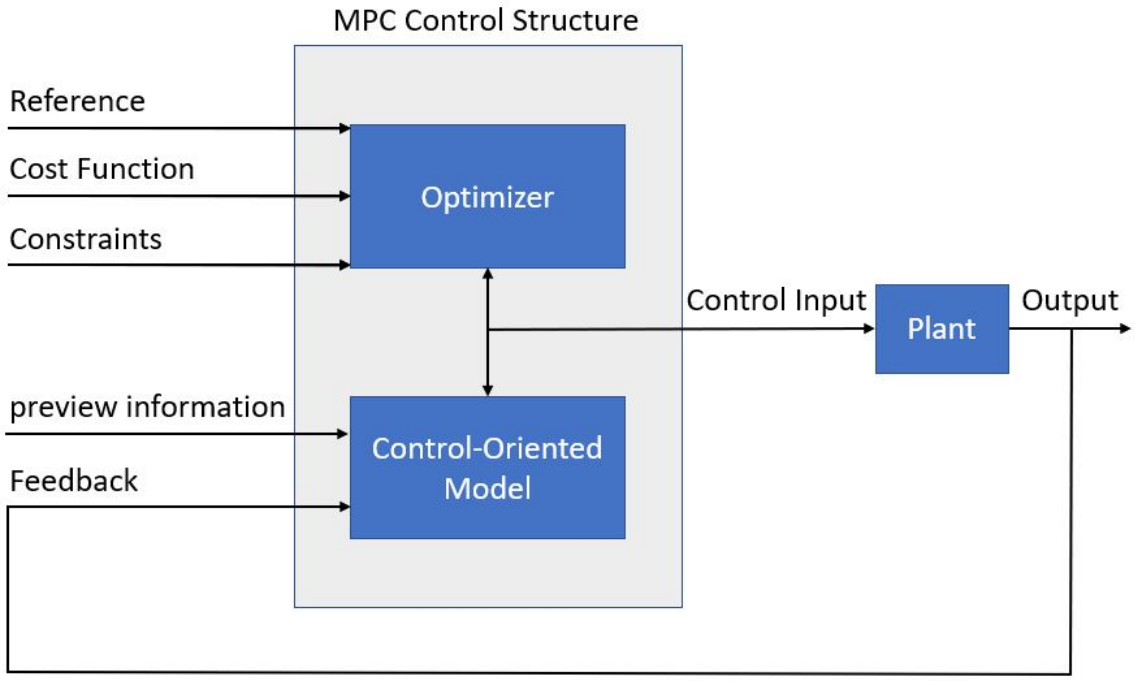

Figure 2.3: Block diagram of MPC [60]

NMPC is a type of MPC uses a nonlinear model as the control-oriented model [5, 26]. Similar to MPC, it tries to minimize errors with a minimal efforts. The main issue in this

technique is the huge computational cost of NMPC in a real-time application which also depends on the complexity of the control-oriented model inside the controller. So far, due to the recent improvements in hardware industry, various NMPC-based controllers have been designed to solve the trajectory tracking problem for different platforms [69, 70]; such as energy management systems [12, 54], automotive systems [4, 10, 29, 59], fixed-wing UAVs [21, 34], quadrotors, etc. Explicit MPC (eMPC) is one of the solutions to the real-time implementation problem suggested in the literature [48]. Instead of finding the optimum solution at each time-step online, this method solves the optimization problem offline and saves the optimal values as a look-up table which can be applied in the heart of NMPC.

Besides, different approaches have been investigated to apply the NMPC to quadcopters. A switching model predictive controller is introduced and simulated in [3]. The NMPC method is used to control a human-sized quadrotor [71]. These approaches have been all used to act as a low-level controller, while another alternative is to design a two-level control architecture for drones with an NMPC-based high-level controller which is presented in this study.

Due to the computational cost of NMPC in real-time optimization process, the use of fast optimizers is essential to implement this approach in practice. As a result, in previous studies, for instance, [23], where a condensed multiple shooting continuation generalized minimal residual (CMSCGMRES)-based NMPC is proposed as a high-level controller for a commercial UAV, significant simplifications have been considered for creating the quadrotor's control-oriented model to make the resulting controller real-time implementable. It should be noted that generalized minimal residual (GMRES) method proposes a fast algorithm which is applying the Krylov subspace technique for solving non-symmetric systems [53, 60].

2.4 Summary

In the past few decades, many studies have investigated to develop a more efficient method to solve the path following problem for quadrotor UAVs. While quadrotors model equations are highly nonlinear, many of these studies designed model-based controllers based on the linear control-oriented models of quadrotors, and it is well-known that these model cannot represent quadrotor's motion appropriately. The most common technique suggested in the literature to overcome the effect of inaccurate control-oriented model is adaptive control approaches. These methods use PI algorithms, such as gradient, LS, etc. to update parameters of linear systems. Also, adaptive approaches were vastly applied to handle uncertainties. However, in contrast to nonlinear models, adaptive models are not accurate

enough. NMPC is an optimal controller which minimizes system errors with minimum efforts, and also, it applies a nonlinear model as the control-oriented model. Although many different nonlinear-based controllers have been suggested in the literature, NMPC is one of the fewest approaches which can manage equality and inequality constraints. On the other hand, NMPC is usually computationally expensive. As a result, a few articles implemented this method as a high-level controller on quadrotor UAVs. To make this controller real-time implementable, these studies used a less realistic simplified nonlinear model. Moreover, NMPC is a feedback-based controller. The performance of this type of controllers highly depends on the accuracy of the measured feedback signal.

Chapter 3

NMPC Design for Quadrotor UAVs

3.1 Introduction

This chapter discusses the NMPC approach in detail. To design this model-based controller, a control-oriented model should be stated. After deriving the model, the Hamiltonian method is introduced as a solution to NMPC's optimization problem. Newton/GMRES is a method which can compute the solution of the Hamiltonian set of equations in real-time application. Newton/GMRES uses fdgmres algorithm which is a fast technique to solve this problem. Furthermore, the designed NMPC is simulated and experimented on a commercial drone, called AR.Drone 2.0 in MATLAB/Simulink. Also, some information about AR.Drone 2.0 and Vicon Vantage motion capture system, which provides some of the feedback signals, is presented.

3.2 Problem Statement

To solve the path following problem for quadrotors, a high-level controller should be designed. As discussed in Chapter 2, this controller can receive the current pose or path and generate the desired direction and transfer it to the low-level controller. To solve this problem, the model of the system, which shows the effect of input generated by the high-level controller on the quadrotor state, should be defined. It will be shown that this model is highly nonlinear. Therefore, to apply an NMPC technique, a fast algorithm should be used to solve the optimization problem inside of NMPC.

3.3 Control-Oriented Modeling

The free body diagram of the drone is depicted in Figure 3.1. As shown, each rotor produces thrust which are represented by F_1, F_2, F_3, and F_4. Accordingly, the resultant force, F, can be expressed as

$$F = F_1 + F_2 + F_3 + F_4 \tag{3.1}$$

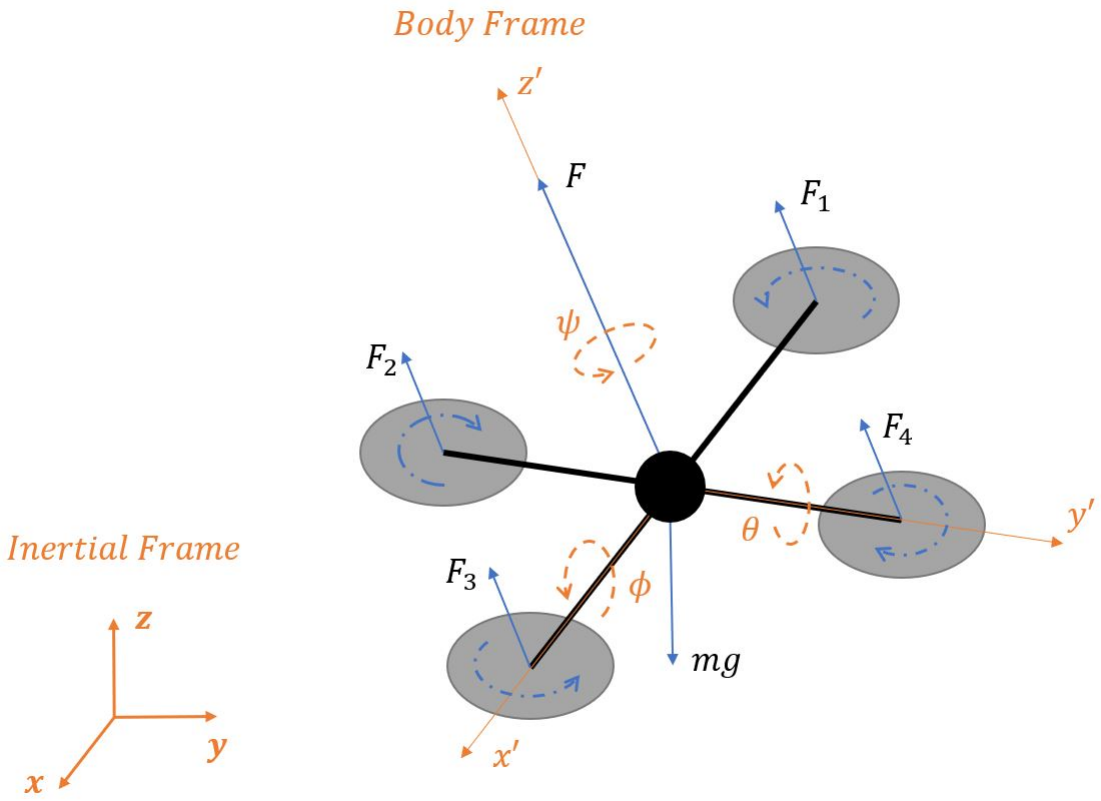

Figure 3.1: Free body diagram of quadrotor

Therefore,

$$m \begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{pmatrix} = -m \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix} + R \times \begin{pmatrix} 0 \\ 0 \\ F \end{pmatrix} \tag{3.2}$$

where m, g, and R are the mass of quadrotor, the gravity acceleration, and the rotation matrix between body frame $(x'y'z')$ and earth fixed frame (xyz). Also, the rotation matrix can be denoted as

$$\begin{aligned} R &= R(z) \times R(y) \times R(x) \\ &= \begin{pmatrix} \cos\psi\cos\theta & -\sin\psi\sin\phi + \cos\psi\sin\theta\sin\phi & \sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi \\ \sin\psi\cos\theta & \cos\psi\sin\phi + \cos\psi\sin\theta\sin\phi & -\cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi \\ \sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{pmatrix} \end{aligned} \tag{3.3}$$

where ϕ (roll), θ (pitch), and ψ (yaw) are rotations about x, y, and z axis. Hence, by replacing equation 3.3 in equation 3.2, the following can be defined:

$$m \begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{pmatrix} = -m \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix} + \begin{pmatrix} (\sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi)F \\ (-\cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi)F \\ \cos\theta\cos\phi F \end{pmatrix} \tag{3.4}$$

Thus from equation 3.4, the following can be written:

$$m\ddot{z} = -mg + \cos\theta\cos\phi F \tag{3.5}$$

Therefore, F can be expressed as a function of $\ddot{z}$.

$$F = \frac{m(\ddot{z} + g)}{\cos\theta\cos\phi} \tag{3.6}$$

Consequently, by using equation 3.6, equation 3.4 can be expressed as

$$\begin{aligned} m\ddot{x} &= (\sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi)\frac{m(\ddot{z} + g)}{\cos\theta\cos\phi} \\ m\ddot{y} &= (-\cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi)\frac{m(\ddot{z} + g)}{\cos\theta\cos\phi} \end{aligned} \tag{3.7}$$

As a result,

$$\begin{aligned} \ddot{x} &= (\frac{\sin\psi\tan\phi}{\cos\theta} + \cos\psi\tan\theta)(\ddot{z} + g) \\ \ddot{y} &= (-\frac{\cos\psi\tan\phi}{\cos\theta} + \sin\psi\tan\theta)(\ddot{z} + g) \end{aligned} \tag{3.8}$$

As discussed, the inputs of the plant which are the outputs of the high-level controller are: $\mathbf{u} = [u_1, u_2, u_3, u_4]^T = [\dot{z}, \phi, \theta, \dot{\psi}]^T$. By applying $\dot{z}$ as u_1, the term $\dot{u}_1$ will appear in

the difference equations. Therefore, equation 3.8 can be denoted by

$$\begin{aligned} \ddot{x} &= (\frac{\sin\psi\tan u_2}{\cos u_3} + \cos\psi\tan u_3)(\dot{u}_1 + g) \\ \ddot{y} &= (-\frac{\cos\psi\tan u_2}{\cos u_3} + \sin\psi\tan u_3)(\dot{u}_1 + g) \end{aligned} \tag{3.9}$$

As known the is in the form of $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$. However, using equation 3.9 leads to the form of $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, \dot{\mathbf{u}})$. Hence, to solve this issue, u_1 is defined as $\ddot{z}$:

$$\mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} = \begin{pmatrix} \ddot{z} \\ \phi \\ \theta \\ \dot{\psi} \end{pmatrix} \tag{3.10}$$

and $\dot{z}$ will be one of the states.

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} = \begin{pmatrix} x \\ \dot{x} \\ y \\ \dot{y} \\ z \\ \dot{z} \\ \psi \end{pmatrix} \tag{3.11}$$

As a result, in this way, the differential equations can be denoted as

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \end{pmatrix} = \begin{pmatrix} \dot{x} \\ \ddot{x} \\ \dot{y} \\ \ddot{y} \\ \dot{z} \\ \ddot{z} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} \dot{x} \\ (\frac{\sin\psi\tan\phi}{\cos\theta} + \cos\psi\tan\theta)(\ddot{z} + g) \\ \dot{y} \\ (-\frac{\cos\psi\tan\phi}{\cos\theta} + \sin\psi\tan\theta)(\ddot{z} + g) \\ \dot{z} \\ \ddot{z} \\ \dot{\psi} \end{pmatrix} \tag{3.12}$$

and the system model can be obtained as

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) = \begin{pmatrix} x_2 \\ (\frac{\sin x_7\tan u_2}{\cos u_3} + \cos x_7\tan u_3)(u_1 + g) \\ x_4 \\ (-\frac{\cos x_7\tan u_2}{\cos u_3} + \sin x_7\tan u_3)(u_1 + g) \\ x_6 \\ u_1 \\ u_4 \end{pmatrix} \tag{3.13}$$

As shown in equation 3.13, the system model is highly nonlinear. Therefore, a nonlinear model-based control is needed to be taken into account. The output of any controller designed based on this model is the desired $\ddot{z}$ (u_1), while as discussed, $\dot{z}$ is the input of the plant. Therefore, an integrator must be applied to convert the desired $\ddot{z}$ coming from the controller to the desired $\dot{z}$, and send this new value to the quadrotor's micro-controller.

3.4 Control Approach

Using the control-oriented model of the system presented in section 3.3, the NMPC problem can be formulated as follows:

$$\begin{aligned} &Minimize \quad J = \frac{1}{2}\sum_{i=0}^{N-1}\left(\left(\mathbf{x}_i\left(t\right)-\mathbf{x}_d\left(t\right)\right)^T\mathbf{q}\left(\mathbf{x}_i\left(t\right)-\mathbf{x}_d\left(t\right)\right)+\left(\mathbf{u}_i\left(t\right)\right)^T\mathbf{r}\left(\mathbf{u}_i\left(t\right)\right)\right)\Delta\tau \\ &Subject\ to: \quad \begin{cases} \mathbf{x}_{i+1}(t) = \mathbf{x}_i(t) + f\left(\mathbf{x}_i\left(t\right),\mathbf{u}_i\left(t\right)\right)\Delta\tau \\ g(\mathbf{x}_i(t),\mathbf{u}_i(t)) = 0 \\ C(\mathbf{x}_i(t),\mathbf{u}_i(t)) < 0 \end{cases} \end{aligned} \tag{3.14}$$

where $\Delta\tau$ is the stepping time, N denotes the number of prediction horizon steps, $f(.)$ is the dynamics of the system, $g(.)$ refers to equality constraints, and $C(.)$ expresses inequality constraints.

Similar to LQR, the purpose of NMPC is to select u(t) to minimize the errors between the actual and desired states with minimal effort. However, they have major differences. Firstly, the NMPC's cost function can take different forms other than a quadratic function. Secondly, constraints can be added to the NMPC problem definition such that keep the system's states, outputs, or inputs within specific boundaries.

The discretization method used to find the discrete model is the forward Euler which is expressed as

$$\dot{\mathbf{x}}(i\Delta\tau) \cong \frac{\mathbf{x}_{i+1}(t)-\mathbf{x}_i(t)}{\Delta\tau} = f\left(\mathbf{x}_i\left(t\right),\mathbf{u}_i\left(t\right)\right) \tag{3.15}$$

Then,

$$\mathbf{x}_{i+1}(t) = \mathbf{x}_i(t) + f\left(\mathbf{x}_i\left(t\right),\mathbf{u}_i\left(t\right)\right)\Delta\tau \tag{3.16}$$

In contrast to this algorithm, other discretization methods such as Runge-Kutta, are much more complicated. Solving NMPC's problem in a real-time application is usually expensive. Therefore, employing a fast algorithm to approximate the discrete model is important.

Also, techniques more robust and stable like backward Euler is cannot be applied, due to the need for future data while is not available.

NMPC optimization problems can be solved by Hamiltonian and Newton/GMRES methods [35, 47, 60]. Based on the Hamiltonian method for a more general case as follows:

$$\begin{aligned} &Minimize \quad J = \Phi(\mathbf{x}_N(t), \mathbf{p}_N(t)) + \sum_{i=0}^{N-1} L(\mathbf{x}_i(t), \mathbf{u}_i(t), \mathbf{p}_i(t))\Delta\tau \\ &Subject\ to: \quad \begin{cases} \mathbf{x}_{i+1}(t) = \mathbf{x}_i(t) + f\left(\mathbf{x}_i\left(t\right), \mathbf{u}_i\left(t\right), \mathbf{p}_i(t)\right)\Delta\tau \\ g(\mathbf{x}_i(t), \mathbf{u}_i(t), \mathbf{p}_i(t)) = 0 \\ C(\mathbf{x}_i(t), \mathbf{u}_i(t), \mathbf{p}_i(t)) < 0 \end{cases} \end{aligned} \tag{3.17}$$

the Hamiltonian, H, denoted by

$$H(\mathbf{x}, \lambda, \mathbf{u}, \nu, \mathbf{p}) = L(\mathbf{x}, \mathbf{u}, \mathbf{p}) + \lambda^T L(\mathbf{x}, \mathbf{u}, \mathbf{p}) + \nu^T g(\mathbf{x}, \mathbf{u}, \mathbf{p}) \tag{3.18}$$

in which $\mathbf{p}_i(t)$ expresses a vector of given time-dependent parameters; λ and ν denote the co-states and Lagrange multipliers, respectively.

The optimum value of input is the solution of the following set of equations proved in [37, 45]:

$$\begin{cases} \mathbf{x}_{i+1}(t) = \mathbf{x}_i(t) + f\left(\mathbf{x}_i\left(t\right), \mathbf{u}_i\left(t\right), \mathbf{p}_i(t)\right)\Delta\tau \\ \lambda_i(t) = \lambda_{i+1}(t) + H_x^T(\mathbf{x}_i(t), \lambda_{(i+1)}(t), \mathbf{u}_i(t), \nu_i(t), \mathbf{p}_i(t))\Delta\tau \\ H_u(\mathbf{x}_i(t), \lambda_{(i+1)}(t), \mathbf{u}_i(t), \nu_i(t), \mathbf{p}_i(t)) = 0 \\ g(\mathbf{x}_i(t), \mathbf{u}_i(t), \mathbf{p}_i(t)) = 0 \end{cases} \tag{3.19}$$

where $\mathbf{x}_0(t)$ is the current state provided by feedback signal, and

$$\lambda_N(t) = \Phi_{\mathbf{x}}^T(\mathbf{x}_N(t), \mathbf{p}_N(t)) \tag{3.20}$$

Therefore, by calculating the states forward in time and the co-states backward in time from the first and second equations of equation 3.19, $\mathbf{x}_i$ and λ_i can be found as sequences of $U(t) = [\mathbf{u}_0^T, \nu_0^T, \mathbf{u}_1^T, \nu_1^T, \ldots, \mathbf{u}_{(N-1)}^T, \nu_{(N-1)}^T]$. Then, the rest of equation 3.19 can be written as one equation denoted by $F(U, x, t)$:

$$F(U, x, t) = \begin{pmatrix} H_u(\mathbf{x}_0(t), \lambda_1(t), \mathbf{u}_0(t), \nu_0(t), \mathbf{p}_0(t)) \\ g(\mathbf{x}_0(t), \mathbf{u}_0(t), \mathbf{p}_0(t)) \\ H_u(\mathbf{x}_1(t), \lambda_2(t), \mathbf{u}_1(t), \nu_1(t), \mathbf{p}_1(t)) \\ g(\mathbf{x}_1(t), \mathbf{u}_1(t), \mathbf{p}_1(t)) \\ \vdots \\ H_u(\mathbf{x}_{N-1}(t), \lambda_N(t), \mathbf{u}_{N-1}(t), \nu_{N-1}(t), \mathbf{p}_{N-1}(t)) \\ g(\mathbf{x}_{N-1}(t), \mathbf{u}_{N-1}(t), \mathbf{p}_{N-1}(t)) \end{pmatrix} = 0 \tag{3.21}$$

To solve equation 3.21, Newton's method will be used, namely;

$$F_U\left(U^k\left(t\right), x^k\left(t\right), t\right)\delta U\left(t\right) = -F\left(U^k\left(t\right), x^k\left(t\right), t\right) \tag{3.22}$$

$$U^{(k+1)}(t) = U^k(t) + \delta U(t) \tag{3.23}$$

Due to the complexity of calculation of Jacobian of F_U, Newton/GMRES method involves the use of fdgmres technique which approximates the Jacobian of F_x by a forward difference approximation:

$$F_x(x) \approx \frac{F(x+hw) - F(x)}{h} \tag{3.24}$$

Although by using Newton/GMRES method some approximations are made, the NMPC optimization problem can be solved quickly while the result will be accurate enough.

The algorithm of fdgmres for finding Newton's step (δU) is represented in [35, 60], as follows:

Algorithm 1: fdgmres algorithm to solve δU [35, 60]

Result: $\delta U = fdgmres(\delta U, x, p, F, k_{max}, \eta, \rho)$
$\delta U = 0$, $r = F(U, x, t)$, $\upsilon_1 = r/||r||$, $\beta = \rho$, $k = 0$;
while $\rho > \eta||F(U, x, t)||$ *and* $k < k_{max}$ **do**
  $k = k + 1$;
  $\upsilon_k + 1 = D_h F(U, x, t : \upsilon_k, 0, 0)$;
  for $j = 1, 2, ..., k$ **do**
    $h_{j,k} = \upsilon_{k+1}^T \upsilon_j$;
    $\upsilon_{k+1} = \upsilon_{k+1} - h_{j,k}\upsilon_j$;
  end
  $h_{k+1,k} = ||\upsilon_{k+1}||$;
  $\upsilon_{k+1} = \upsilon_{k+1}/||\upsilon_{k+1}||$;
  $e_1 = (1, 0, ..., 0)^T \in \mathbb{R}^{k+1}$;
  $H_k = [h_{ij}] \in \mathbb{R}^{(k+1)\times k}$ (if $i > j + 1$ then $h_{ij} = 0$);
  Minimize $||\beta e_1 - H_k y^k||$ to find $y^k \in \mathbb{R}^k$;
  $\rho = ||\beta e_1 - H_k y^k||$;
  $V_k = [\upsilon_i] \in \mathbb{R}^{mN\times k}$;
end
$\delta U = V_k y^k$;

based on equation 3.23, to find the optimum input, δU can be added to the previous implemented U. Then the new input can be applied to the system.

In practice, after solving this optimization problem and applying the calculated optimum input, the controller checks the inequality equations ($C(\mathbf{x}_i(t), \mathbf{u}_i(t)) < 0$) to find out whether they are satisfied. If the input can satisfy the inequality constraints, then it will be applied to the system. On the other hand, if these conditions will not be satisfied ($C(\mathbf{x}_i(t), \mathbf{u}_i(t)) \geqslant 0$), the NMPC should find another optimum input which can satisfy them. In this case, equation 3.19 will be changed to:

$$\begin{cases} \mathbf{x}_{i+1}(t) = \mathbf{x}_i(t) + f\left(\mathbf{x}_i\left(t\right), \mathbf{u}_i\left(t\right), \mathbf{p}_i(t)\right)\Delta\tau \\ \lambda_i(t) = \lambda_{i+1}(t) + H_x^T(\mathbf{x}_i(t), \lambda_{(i+1)}(t), \mathbf{u}_i(t), \nu_i(t), \mathbf{p}_i(t))\Delta\tau \\ H_u(\mathbf{x}_i(t), \lambda_{(i+1)}(t), \mathbf{u}_i(t), \nu_i(t), \mathbf{p}_i(t)) = 0 \\ g(\mathbf{x}_i(t), \mathbf{u}_i(t), \mathbf{p}_i(t)) = 0 \\ C(\mathbf{x}_i(t), \mathbf{u}_i(t), \mathbf{p}_i(t)) - \epsilon = 0 \end{cases} \tag{3.25}$$

where $\epsilon > 0$, and the value of this parameter is chosen close to 0 by the user. Equation 3.25 pushes the optimum input from the global optimum point to one of the first values which satisfies inequality constraints.

3.5 Simulations

The discussed controller has been implemented in the ARDrone Simulink Development Kit V1.1 [55]. The Simulink toolbox contains a simulation environment that simulates the AR.Drone 2.0 which is parameterized by PI techniques. On the other hand, it also provides a connection between the Simulink and AR.Drone 2.0 quadrotor over WiFi which can be employed for practical tests. This communication provides the user with onboard sensors data, and it allows him/her to analyze the feedback data and transfer new commands to the quadrotor.

Also, the MPsee toolbox [60] which is developed in our group's previous works, is used to implement the designed NMPC controller. After modifying the model and the controller's parameters, such as receding horizon length and time steps, N, $\mathbf{q}$, and $\mathbf{r}$, this toolbox generates a Newton/GMRES-based NMPC controller block which is very fast by using the fdgmres algorithm.

Figure 3.2 shows the simulation of the designed NMPC on altitude control. This graph reveals that the NMPC can control the quadrotor in simulation, but it converges very slowly with the percentage overshoot of more than 45%. This unsatisfactory result is mostly because of the feedback signal. This signal should provide observation of all states. However, the vertical velocity is not provided by any measurement equipment and estimating this unknown state will highly improve the performance of altitude control. Moreover,

The delay in tracking the desired value is because of a delay block in the simulation plant which holds the command coming from the controller for 0.26 s. This approach is applied to simulate the time required to send the command from the ground station to the quadrotor.

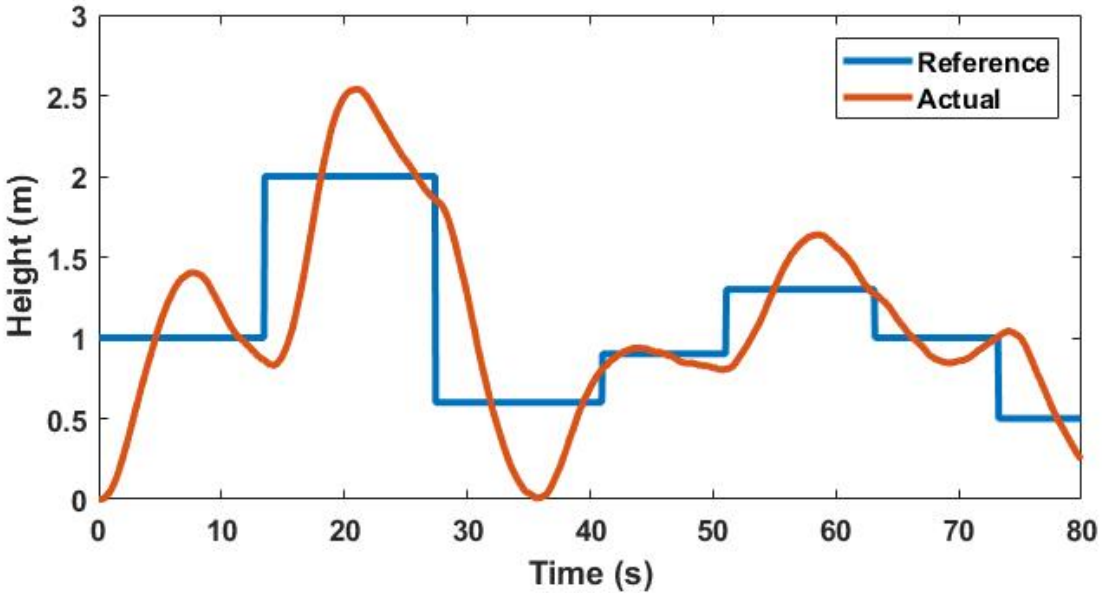

Figure 3.2: Simulation of the designed NMPC on altitude control

3.6 Experiments

In this section, firstly the experiment setup, containing AR.Drone 2.0 and Vicon motion capture system, is explained, and then the experimental result is represented.

3.6.1 Experiment Setup

AR.Drone 2.0

The proposed controller and observers are applied to AR.Drone 2.0 (see Figure 3.3). This quadrotor is developed by a French company, called Parrot Drones SAS. This drone is equipped by a micro-controller which uses OMAP 3630 CPU based on a 32 bit ARM Cortex

A8 (with a frequency of 1 GHz) [50]. The GPU of this processor is PowerVR SGX530, and it uses a 1 GB DDR2-RAM as memory. This system is running with Kernel Linux uclibc 2.6.32.9-gbb4d210, and it uses uclibc as c-library. Moreover, it can communicate over a wireless local area network (WLAN) to send flight and video data.

Figure 3.3: AR.Drone 2.0

This drone's rotors powered by brushless motors with three phases of current [49]. These rotors are connected to the onboard micro-controller which performs as a low-level controller. The controller can handle simple manoeuvres automatically including taking-off and landing. The micro-controller is capable of adjusting itself to different kinds of connected engines and it always detects rotors situation for protection. For example, if the drone hits any obstacle, the controller can detect it by watching the status of the rotor and it can stop all engines to prevent any more damages. Also, this drone can be set in a low wind drag configuration in outdoor flights, while a hull can be added for indoor application to prevent any damages to rotors and wings in case a rotating propeller encounters any obstacle. The AR.Drone 2.0 uses a charged LiPo battery as the power source. When the battery is fully charged, the voltage is 12.5 Volts, and the drone automatically lands and turn off the whole system if it reduces to 9 Volts.

This quadcopter has different types of sensors for localization. It has an ultrasound telemeter which can be used to determine the altitude. This sensor is a range finder sensor which releases high-frequency sound waves; then the range finder waits for an emitted wave to be reflected. Then, by knowing the sound speed and measuring the time between the releasing and receiving data, the distance between the sensor and obstacle caused reflection

can be estimated. It is noted that sound-based sensors can detect any object, while radar and light-based range finders cannot sense transparent materials properly. On the other hand, ultrasound is unreliable if the object absorbs the sound or the object reflects it away from the receiver. Also, the sound speed depends on temperature and humidity. Therefore, by using ultrasound telemeter, some inaccuracy is unavoidable.

The ultrasound sensor can only detect less than 6 meters in altitude. Therefore, for higher flight height, the AR.Drone is equipped by a pressure altimeter. This sensor can measure the atmospheric pressure and then, estimates the drone's altitude. Moreover, this quadcopter uses a QVGA (320x240) camera aiming towards the ground to provide ground speed measurement. This can be done by image processing. In addition to the mentioned sensors, the drone contains a 3 axis accelerometer, a 3 axis gyroscope, and a 3 axis magnetometer.

Vicon Vantage Motion Capture System

Our test laboratory is equipped with a Vicon Vantage motion capture system including eight cameras (see Figure 3.4). This system can track spherical markers which are attached to the quadrotor and provide their position. The system is interconnected to a ground station computer on a local area network (LAN) and transfers the position data over UDP. Also, the ARDrone Simulink Development Kit V1.1 provides a toolbox which can communicate with AR.Drone's onboard computer via WiFi. The toolbox is installed on the ground station computer. The control and estimation approaches discussed in this study are implemented on this station, and the NMPC and observers can use the UDP data as feedback and send the optimal inputs to the UAV over WiFi. Also, the UAV's onboard computer is capable of transferring a reliable observation of $\dot{x}$ and $\dot{y}$ received from onboard sensors.

Figure 3.4: Experiment environment

The Vicon Vantage motion capture system main components are:

- Vicon Vantage motion capture cameras
- Vicon Vantage networking PoE+ switches
- Vicon Vantage software
- Vicon Vantage host PC

This system also consists of Vantage system cables which connect devices, a Vantage calibration device (Vicon Active Wand V2 IR), and Vantage accessories, such as Vicon retroreflective markers and different types of tapes.

Cameras used in this system are Vicon Vero v1.3. The technical specification of this camera is represented in Table 3.1. This version of Vicon Vero camera can capture actors at the frame rate of 250 FPS with 1.3 MP resolution. Therefore, it can track both fast movements and multiple objects with very low latency. Also, it has two sensors: a thermal sensor and an accelerometer. The thermal sensor monitors the camera's temperature to prevent overheating, while the accelerometer measures the camera's position in real time. These two sensors guarantee the optimal performance of the system.

Each Vicon camera connects to a PoE+ switch via cable and this connection provides both data and power. On the other hand, the PoE+ switch connects to the host PC through another port. This PC runs the Vicon Vantage motion capture software which can be used for processing, visualization, and investigating data captured by the cameras. Moreover, the software used in this research is Tracker 3.7 which is usually used for engineering applications, such as robotics.

Technical Specification	Vicon Vero v1.3
Resolution	1.3MP 1280 * 1024
Frame Rate (Full Frame) FPS	250
Pixel Shutter Type	Global
Lens	Vicon 6-12 mm varifocal
Effective FOV (H x V) deg	W: 60.8 x 50.3 T: 32.7 x 26.4
Strobe	850nm 'IR'
Power Consumption	12W
Connectivity	RJ45/Cat5e
Weight	575g
Camera external dimensions H x W x D	83 x 80 x 135 mm

Table 3.1: Technical Specification of Vicon Vero v1.3

Through the Tracker software, the condition of cameras can be checked. Also, this software can detect spherical markers attached to objects by its especial colour and it represents a virtual 3-dimensional environment which shows the real-time position of multiple markers. To create an object like quadrotor UAV, at least three markers should be attached to the object (five is optimum). Then, the software can detect and track the object by memorizing the especial shape created by markers. In this study, quadrotor's pose updates at a rate of 100 Hz. For real-time applications, however, the software is capable of processing the data at a rate of 400 Hz. Furthermore, The Tracker software supports UDP connection and it can send the drone pose to the ground station via LAN connection.

3.6.2 Experiment Results

The NMPC is tried to implement on AR.Drone 2.0 using ARDrone Simulink Development Kit V1.1 [55]. In this test, Vicon Vantage motion capture system is used to provide x, y, z, and ψ. Also, the onboard micro-processor sends $\dot{x}$ and $\dot{y}$ as feedback signals. As

mentioned, sensors cannot measure the vertical velocity ($\dot{z}$). Therefore, the NMPC, which is a feedback-based controller, cannot control the drone efficiently. Due to the system's instability, this test could not be applied without damaging the drone. Therefore, providing any results without measuring a proper vertical velocity is almost impossible.

3.7 Summary and Remarks

In this chapter, the high-level model of the quadrotor has been developed. This nonlinear model has been applied to the heart of NMPC as the control-oriented model. The NMPC control approach investigates to find the optimum input at each time-step. To do so in a real-time application, Newton/GMRES method has been introduced. This method can solve the NMPC's optimization problem by applying the Hamiltonian technique which derives a set of equations with multiple variables. To solve the Hamiltonian set of equations in a real-time application, the Newton/GMRES method employs fdgmres algorithm. The designed NMPC is simulated and experimented on a commercial drone, called AR.Drone 2.0 using MATLAB/Simulink. The result showed that this feedback-based controller cannot perform efficiently with unknown states. Therefore, it is necessary to develop an estimation method to measure the unknown vertical velocity.

Chapter 4

State Estimation

4.1 Introduction

In this chapter, the importance of state estimation methods, which are capable of estimating unknown states, is explained in the problem statement section. After checking the observability of the system by analyzing whether the observability matrix is full rank or not, this chapter presents two different state estimation techniques, namely; the Kalman filter and the Luenberger observer. Moreover, these estimators and the presented NMPC in chapter 3 are simulated and experimented on a commercial drone, called AR.Drone 2.0 in MATLAB/Simulink, and the effect of the designed estimators on NMPC is discussed. Besides, this control technique is compared with a proportional controller, and the result is represented. The experiment uses the Vicon Vantage motion capture system to provide some of the feedback signals.

4.2 Problem Statement

Similar to LQR and other types of feedback-based controllers, NMPC requires to have access to the states of the system. As stated before, two sensors have been employed to detect the states of the considered UAV, namely; Vicon camera system and onboard IMU. The Vicon motion capture system determines x, y, z, and ψ of the state variables, while the IMU observes $\dot{x}$ and $\dot{y}$. Therefore, based on equation 3.11, $\dot{z}$ is the only state which cannot be measured directly. Because of the Vicon system's noisy data, taking derivatives of z for calculations is inaccurate and cannot represent $\dot{z}$ precisely. In addition, as shown in Figure

2.2, $\dot{z}$ is the first input to the low-level controller. On the other hand, as represented in equation 3.10, $\ddot{z}$ is the first output of the high-level controller. Hence, a method should be employed to estimate $\dot{z}$ based on the inaccurate derivative of z and the output of NMPC which is the reference $\ddot{z}$. To do so, the observability should be checked at the earlier stage.

4.3 Observability

As mentioned earlier, all of the states except $\dot{z}$ are measurable. Therefore, according to the definition of observability, if $\dot{z}$ can be determined by knowing the input and the output of the plant over a finite time, the system is completely observable [30]. In this section, based on the observation of z, the observability of $\dot{z}$ is proved. To do so, a system including z and $\dot{z}$ is introduced.

$$\mathbf{x}' = \begin{pmatrix} x_5 \\ x_6 \end{pmatrix} = \begin{pmatrix} z \\ \dot{z} \end{pmatrix} \tag{4.1}$$

$$\dot{\mathbf{x}}' = \begin{pmatrix} x_6 \\ u_1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \mathbf{x}' + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u_1 \tag{4.2}$$

In equation 4.1, only z can be estimated. Hence,

$$y' = \begin{pmatrix} 1 & 0 \end{pmatrix} \mathbf{x}' \tag{4.3}$$

Therefore, the observability matrix can be represented as

$$O = \begin{pmatrix} C \\ CA \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \begin{pmatrix} 1 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{4.4}$$

Therefore, the observability matrix is full rank. As a result, $\dot{z}$ can be determined from observations of x_5 (z) and u_1 ($\ddot{z}$), and the system is observable.

4.4 Kalman Filter

To measure $\dot{z}$, the derivation of z can be computed at each time-step. Given the noise of z estimated by the Vicon Vantage motion capture system, the derivative of z will not be accurate. Therefore, a Kalman filter is used to obtain a more reliable vertical velocity. Moreover, the first high-level controller output (u_1) is $\ddot{z}$, however, the first low-level

controller input is $\dot{z}$. Hence, the predicted vertical velocity obtained by the Kalman filter based on the system model can be transferred to the low-level controller as a reference signal of $\dot{z}$.

Kalman filter is one of the best approaches for applying Bayes filter [61]. This filter is a method to predict continuous states in linear Gaussian systems. This filter is implementable if the Gaussian function characterizes by the moment parameterization: mean and covariance which are a vector with the highest probability and a positive semi-definite and symmetric matrix representing the order of uncertainty, respectively. To design the Kalman filter, the model should be discrete. Therefore,

$$\dot{x}_6 = u_1 \tag{4.5}$$

As mentioned, NMPC is computationally expensive. Therefore, applying a fast algorithm in other parts is important. As a result, a simple, however fast method is employed to discrete the model in equation 4.5. By using the forward Euler approach, the following can be expressed:

$$\frac{x_6[k] - x_6[k-1]}{\Delta t} = u_1[k-1] \tag{4.6}$$

$$x_6[k] = x_6[k-1] + \Delta t \times u_1[k-1] \tag{4.7}$$

Therefore,

$$A = 1, B = \Delta t, C = 1 \tag{4.8}$$

The Kalman estimator presented in [61] can be stated as

$$\begin{aligned}
&\hat{x}_6^-[k] = A\hat{x}_6[k-1] + Bu_1[k-1] \\
&P^-[k] = AP[k-1]A^T + Q \\
&\hat{x}_6[k] = \hat{x}_6^-[k] + K[k](\dot{z}[k] - C\hat{x}_6^-[k]) \\
&K[k] = P^-[k]C^T(CP^-[k]C^T + R)^{(-1)} \\
&P[k] = (I - K(k)C)P^-[k]
\end{aligned} \tag{4.9}$$

where $\hat{x}_6^-$ and $\hat{x}_6$ are the predicted and corrected means, $\dot{z}$ is the derivative of the measured z at each time-step, K is the Kalman gain, and Q and R are the covariances of the process and measurement noises which are assumed to be white Gaussian noises with zero means. Also, P^- and P are the predicted and corrected covariances indicating the accuracy of the calculated means. It should be mentioned that the first two lines of equation 4.9 are called prediction step, while the rest is the correction step. Moreover, the estimated corrected mean considered as the vertical velocity at each time-step. An important assumption to

obtain equation 4.9 is that the process and measurement noises are completely independent. Therefore, while the model and sensor's uncertainties are not directly related, these noises considered as orthogonal signals. As a result, these white noises are crossed-correlated.

To calculate R, a sample of z derivatives is generated and it is compared with the actual vertical velocity as depicted in Figure 4.1. Therefore, R can be estimated by determining the covariance of the difference between the two graphs. The actual velocity is estimated by calculating the smoothed z derivatives. Also, as shown in Figure 4.2, the predicted vertical velocity which can be obtained using the discrete system model can be compared with the true value of $\dot{z}$ for defining Q. Although the discrepancy between these two graphs is small, it shows that this model cannot represent the system even in the simulation environment in the most accurate way. The reason for this discrepancy is that in developing this model, uncertainties or delays were not considered, and it is well-known that these components usually play an important role in every system.

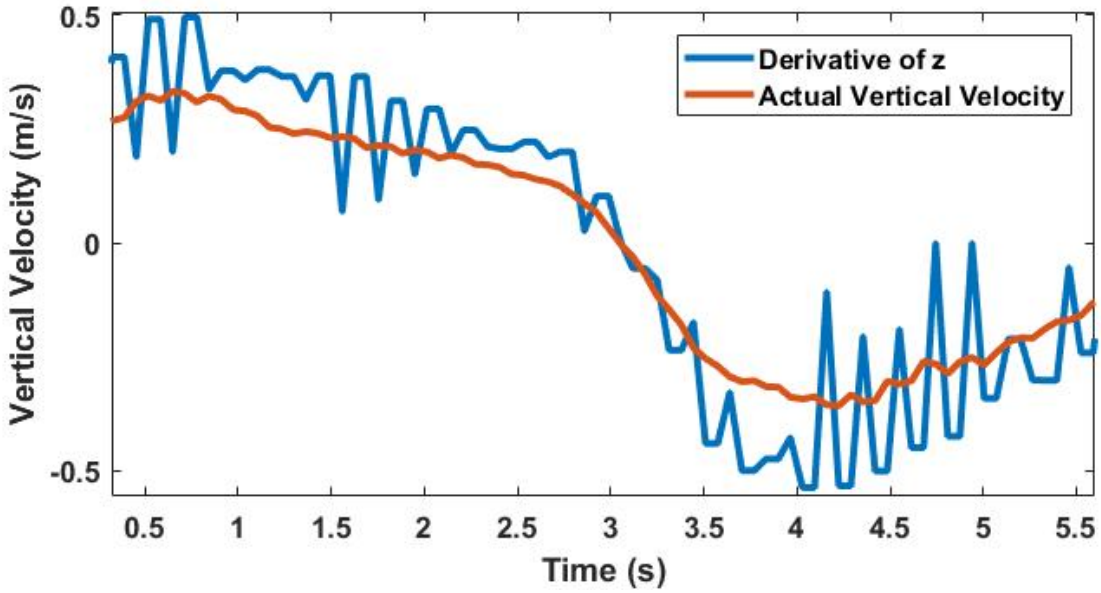

Figure 4.1: Calculating the covariance of the measurement noises

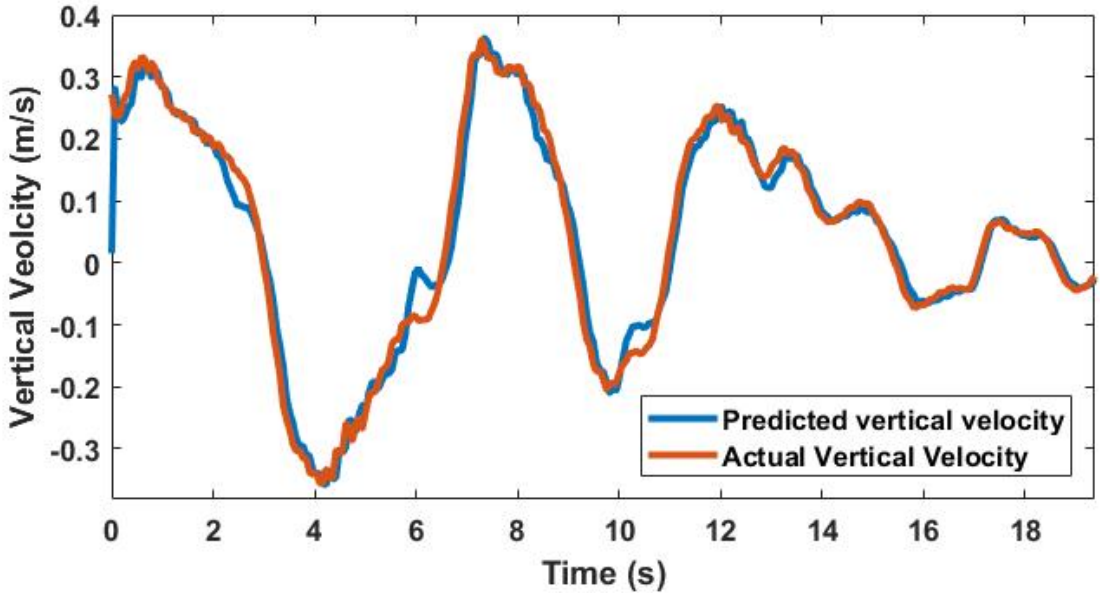

Figure 4.2: Calculating the covariance of the process noises

Figure 4.1 and 4.2 show that the model can estimate the vertical velocity more accurate than the sensor measurement. Therefore, the value of R is much greater than the value of Q. For the above sample, R and Q are calculated as 3.2×10^{-3} and 2.8559×10^{-4}, respectively. This shows that the Kalman filter's estimation mostly relies on the prediction step, and the model employed in the heart of the filter is more reliable than the derivative of the data coming from the Vicon Vantage motion capture system.

4.5 Luenberger Observer

Luenberger observer is another method to determine unknown states. For linear systems like

$$\begin{aligned} \dot{x} &= Ax + Bu + d, \quad x(0) = x_0, \\ y &= Cx \end{aligned} \tag{4.10}$$

where d shows uncertainty and inaccuracy of the model. The Luenberger observer can be designed as follows:

$$\begin{aligned} \dot{\hat{x}} &= A\hat{x} + Bu + K_o(y - \hat{y}), \quad \hat{x}(0) = \hat{x}_0, \\ \hat{y} &= C\hat{x} \end{aligned} \tag{4.11}$$

where for estimating vertical velocity based on the altitude A, B, and C can be denoted as

$$A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \ , \quad B = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \ , \quad C^T = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

and K_o is chosen such that $A - K_oC$ is a stable matrix which means

$$\begin{aligned} &\det\left(sI - (A - K_oC)\right) = (s - \lambda_0)^2 = 0, \\ &\lambda_0 > 0 \end{aligned} \tag{4.12}$$

It is noted that observability guarantees existense of K_o.

The idea behind of this method is that if $A - K_oC$ is a stable matrix, then, as time goes to infinity, the difference between the actual state (x) and the estimated state ($\hat{x}$) converges to zero [30]. To prove it, equation 4.10 subtracted from equation 4.11 gives

$$(\dot{\hat{x}} - \dot{x}) = (A - K_oC)(\hat{x} - x) - d \tag{4.13}$$

by considering $\tilde{x} = \hat{x} - x$, then

$$\dot{\tilde{x}} = (A - K_oC)\tilde{x} - d \tag{4.14}$$

Therefore, due to the accuracy of the model (a sample of it is represented in Figure 4.2) and small covariance of d, while equation 4.12 is true, then $\tilde{x} \to 0$ exponentially as $t \to \infty$.

Practically, the optimum value of λ_0 is found by try and error to be 5. Therefore, based on the equation 4.12, K_o can be found as

$$K_o = \begin{pmatrix} 10 \\ 25 \end{pmatrix} \tag{4.15}$$

Figure 4.3 represents the effect of the value of λ_0 on the estimated velocity by Luenberger observer. The value of this parameter can be increased to have more exponential stability. On the other hand, the accuracy may be compromised because of the aggressiveness of the estimator.

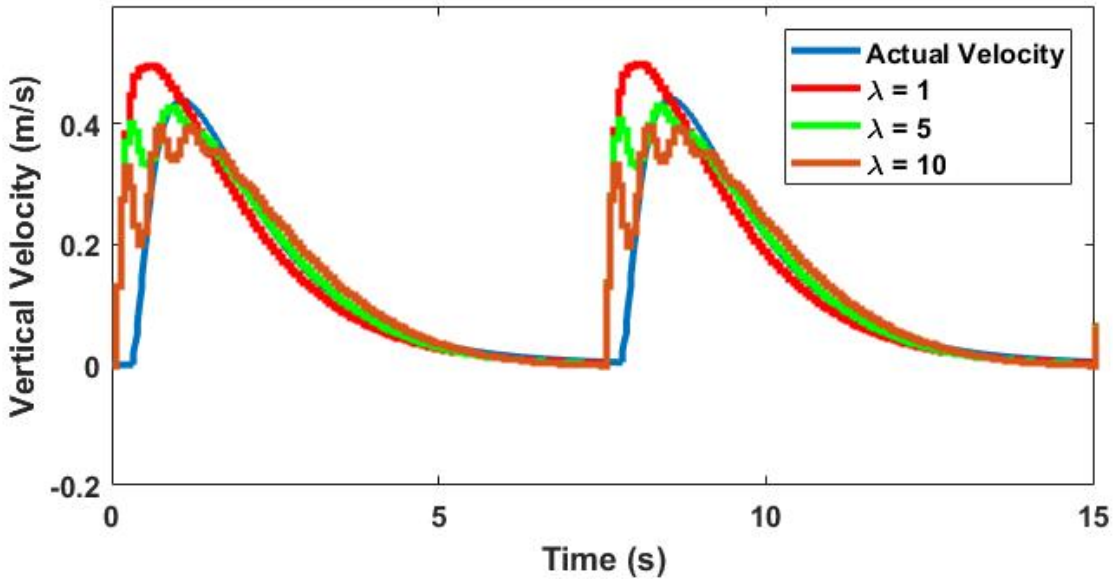

Figure 4.3: Effect of λ_0 on the performance of Luenberger observer

4.6 Simulations

The discussed controller and observers have been simulated using ARDrone Simulink Development Kit V1.1 [55] and MPsee toolbox [60]. Figure 4.4 illustrates the estimated vertical velocity obtained by the Kalman filter, and Figure 4.5 depicts this state estimated by Luenberger observer. As these graphs show, both observers can estimate the actual velocity accurately. The performance of both observers is highly reliable at lower velocities, however, the error of the estimated state grows as velocity increases. Also, Figure 4.6 depicts the effect of these two observers on the performance of NMPC. As depicted in this Figure, the NMPC controller, which needs vertical velocity as a feedback signal from an estimator, can control the UAV platform at different height set-points properly without any overshoot. Also, these graphs show that these two observers have a similar improvement in the control approach.

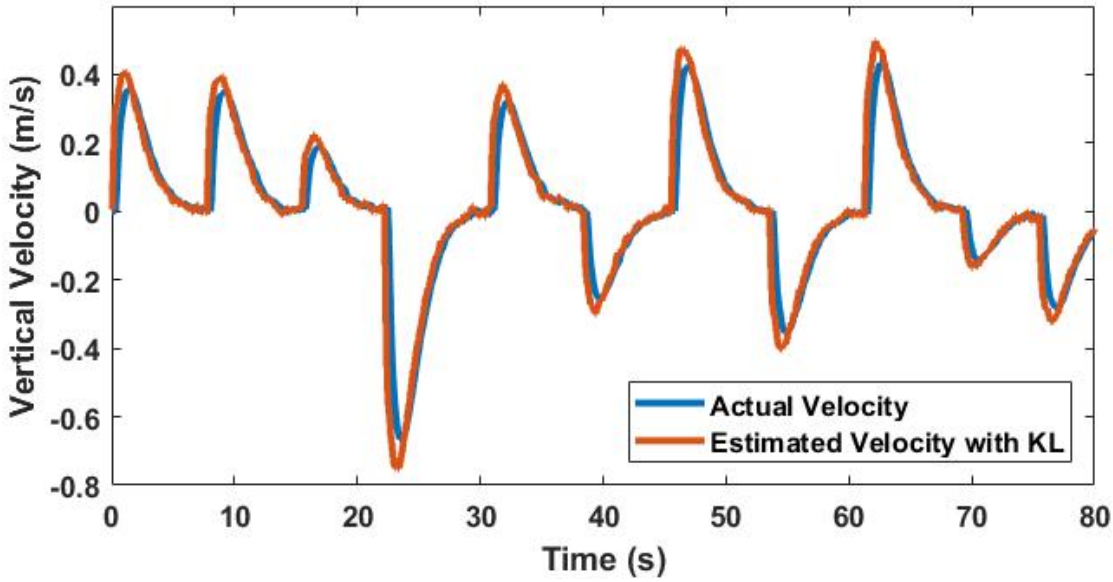

Figure 4.4: Estimated vertical velocity by Kalman filter in simulation

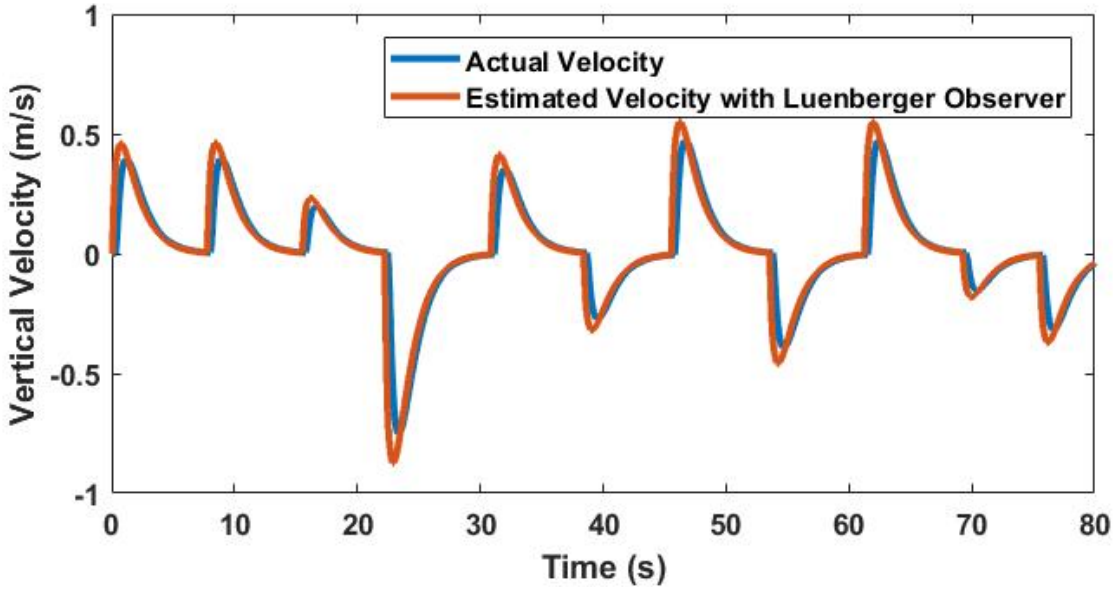

Figure 4.5: Estimated vertical velocity by Luenbereger observer in simulation

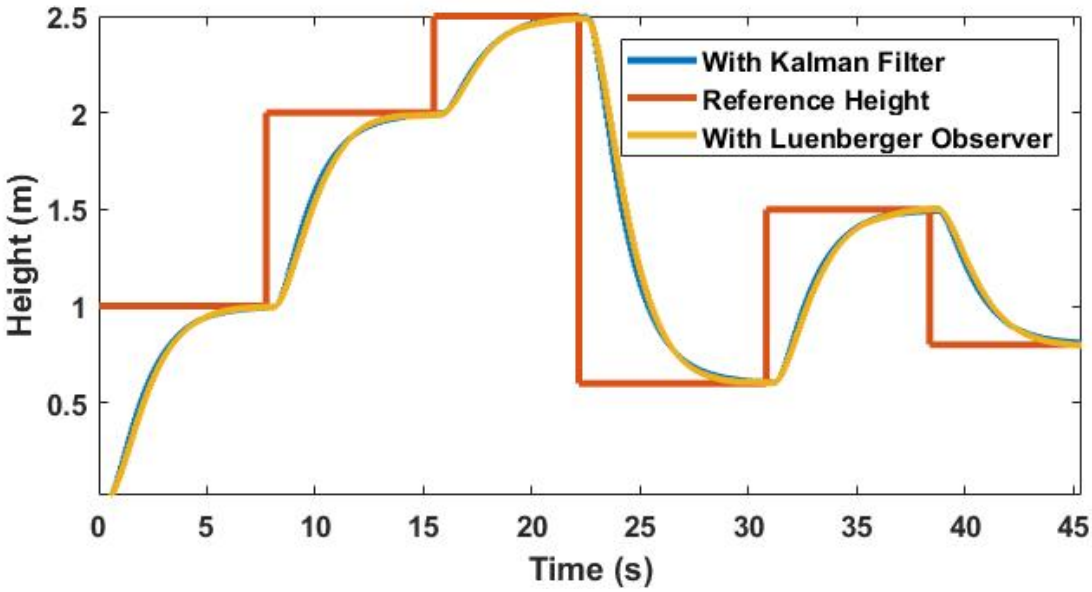

Figure 4.6: Simulation of the height control with applying Luenberger observer and Kalman filter

Also, in Figure 4.7, the result of the designed NMPC before applying these observers represented in Figure 3.2 is compared with the NMPC while using the discussed Kalman filter. This graph shows the effect of estimators, and it proves that applying these estimators will greatly improve the performance of the altitude controller. The actual altitude will converge to the desired heights quickly without any overshoot.

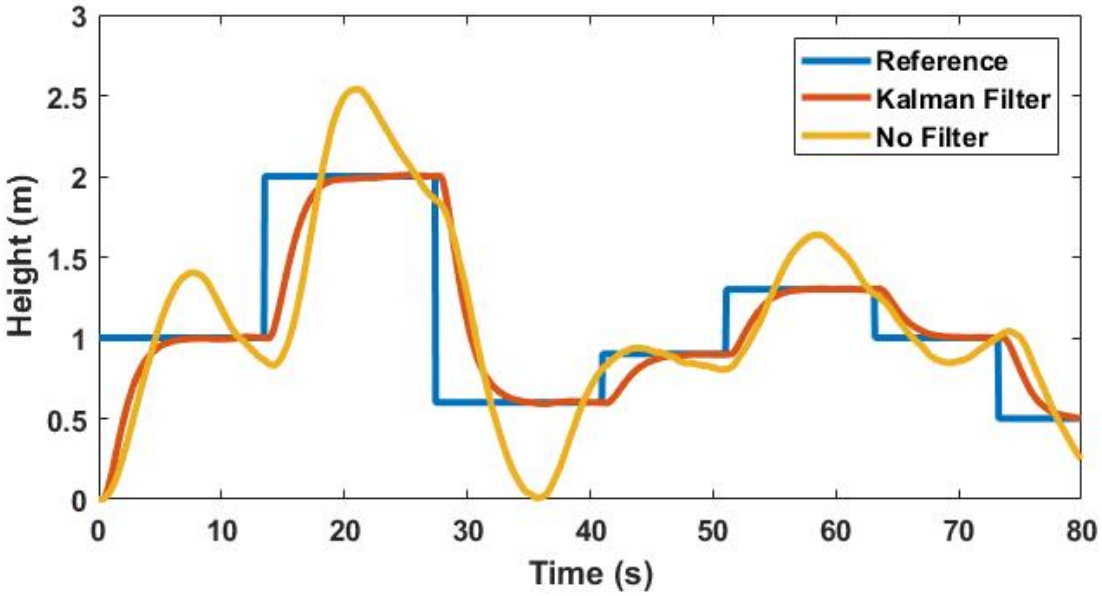

Figure 4.7: The effect of Kalman filter on altitude control

Moreover, the proposed NMPC approach is compared with a proportional controller designed in the ARDrone Simulink Development Kit V1.1 [55]. The formulation of this controller can be expressed as:

$$\begin{aligned} \dot{z} &= k_1(z_d - z) \\ \phi &= k_2(\dot{y}_d - \dot{y}), \quad \dot{y}_d = y_d - y \\ \theta &= k_3(\dot{x}_d - \dot{x}), \quad \dot{x}_d = x_d - x \\ \psi &= k_4(\psi_d - \psi) \end{aligned} \tag{4.16}$$

The simulation results are given in Figure 4.8, 4.9, 4.10, and 4.11. In these graphs, the NMPC method tries to converge to the reference input smoothly while with the proportional controller, the UAV behaves much more aggressively and usually with overshoot. However, while the proportional controller performs better for altitude control, NMPC leads to a smooth response with less overshoots for other coordinates. Although the aggressiveness of the proportional increases the overshoot, the result shows that it reduces the settling time.

In Figure 4.8 (x controller), the settling time for NMPC and proportional controllers are almost the same. It takes around 4 s for both to move the drone from $x = 0$ m

to $x = -1.5\ m$, while NMPC can do the same without any overshoot. Figure 4.9 (y controller) shows that the time required for proportional technique to push the quadrotor for 1 m is 4.5 s, and for NMPC it is 5.5 s. However, the advantage of NMPC is that it can manage to control the quadrotor on y axis with less overshoot. As mentioned, the only advantage of proportional controller is during the height control (see Figure 4.10). For example this approach can make the drone to take off to the 1 m altitude in 2 s with almost no overshoot, while NMPC manages to do so in 5 s. Also, in Figure 4.11 in which the result of heading controller is represented, the settling time for proportional to change the drone's heading from 0 rad to 1 rad is 2.5 s, and the NMPC can change it in 5 s, however with no overshoot.

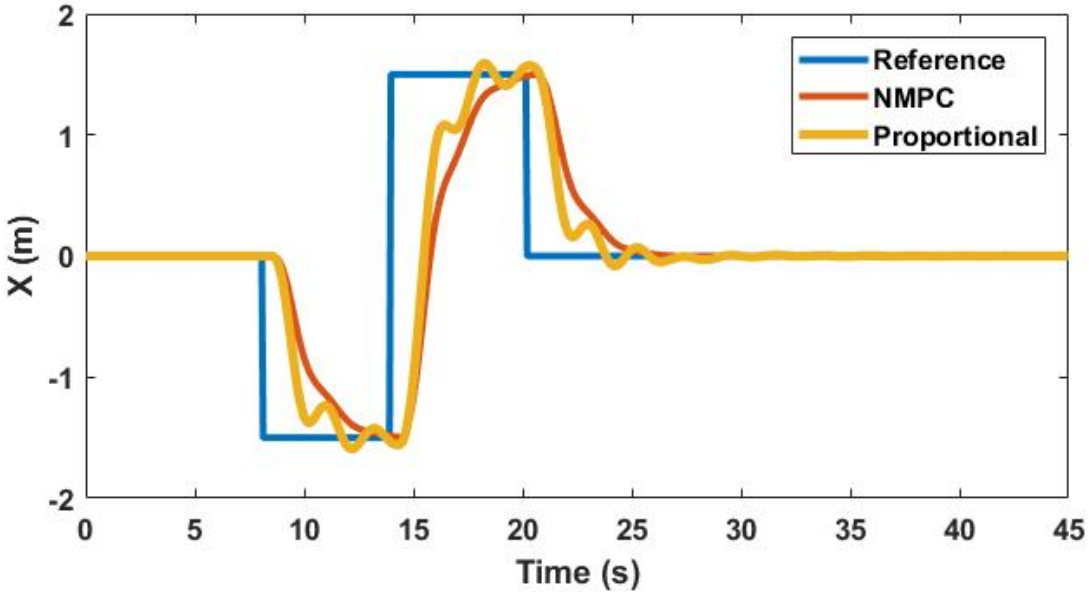

Figure 4.8: NMPC vs. proportional (x controller)

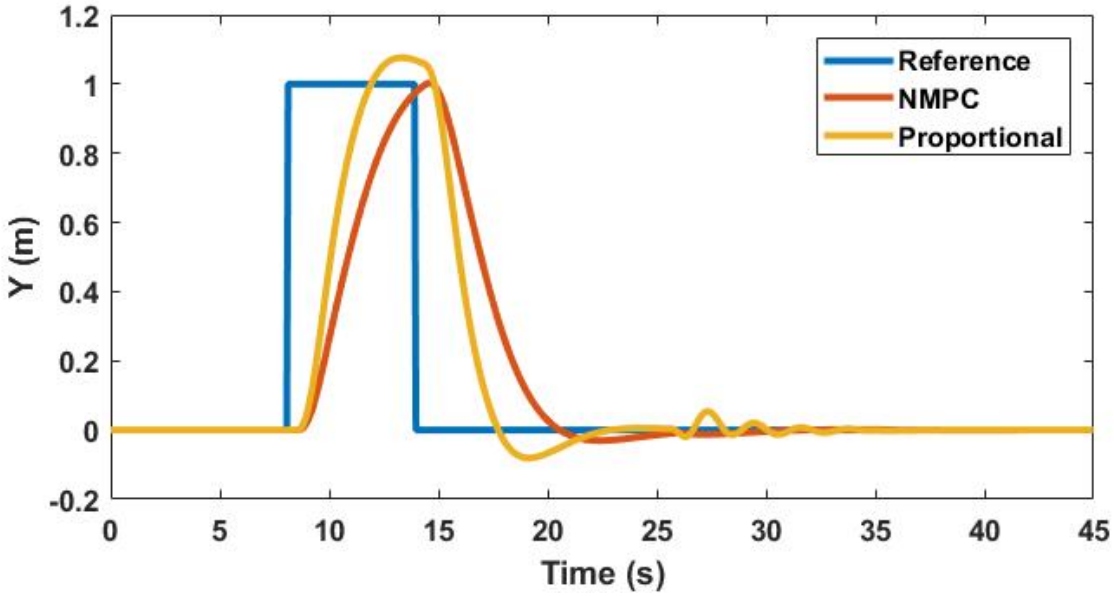

Figure 4.9: NMPC vs. proportional (y controller)

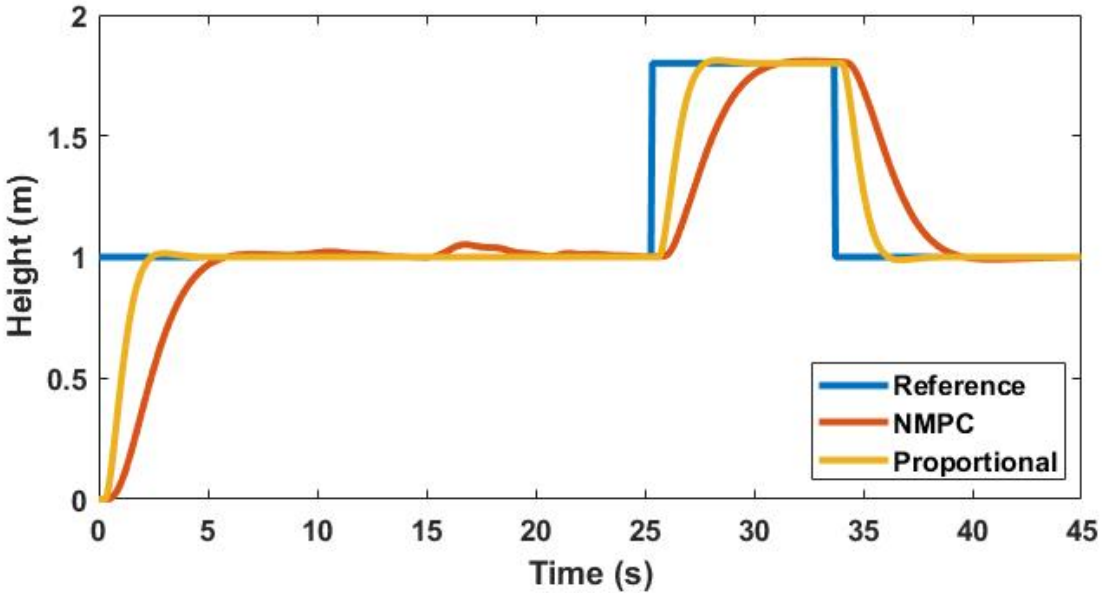

Figure 4.10: NMPC vs. proportional (z controller)

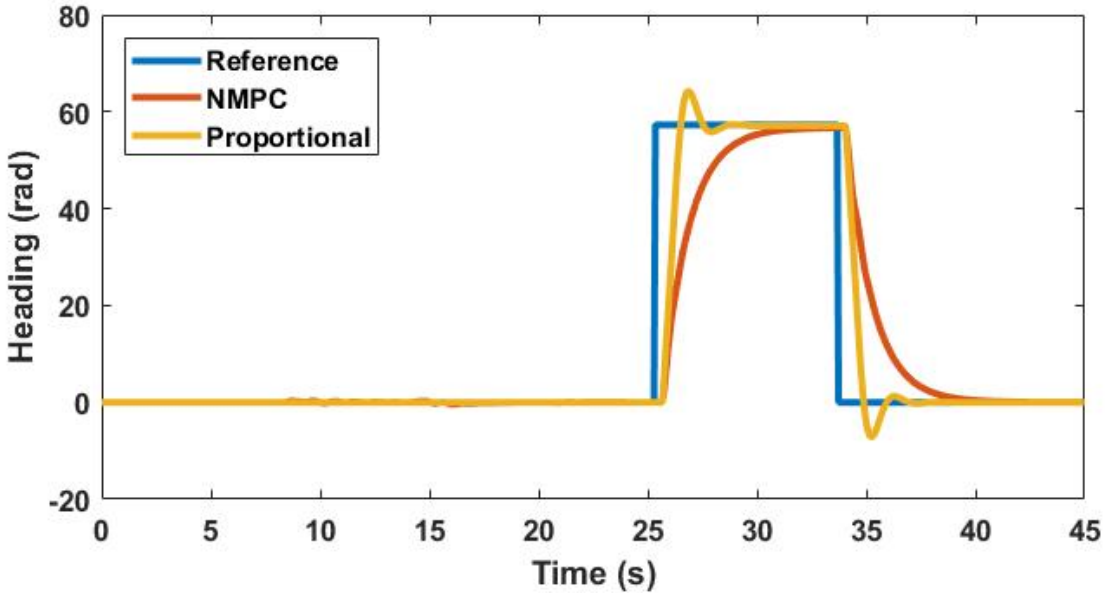

Figure 4.11: NMPC vs. proportional (ψ controller)

As discussed in chapter 3, the NMPC method can take into account both equality and inequality constraints directly, while many other methods like PID are not capable of this. Also, in order to stabilize nonlinear systems, inputs of the system are usually bounded within certain limits. Therefore, in the NMPC problem definition, the following inequality constraints were considered to help with the system stability:

$$-1 < u_1, u_2, u_3, u_4 < 1 \tag{4.17}$$

Equation 4.17 keeps the reference roll and pitch angles between -1 *rad* and 1 *rad*, the reference vertical velocity between -1 $^{m}/_{s}$ and 1 $^{m}/_{s}$, and the reference yaw rate between -1 $^{rad}/_{s}$ and 1 $^{rad}/_{s}$.

Figure 4.12 represents that NMPC can manage this inequality constraint easily while in Figure 4.13 the proportional inputs are much greater. It should be noted that the quadrotor will be unstable if the roll and pitch angles go beyond $\pi/2$ and $-\pi/2$, and proportional inputs are sometimes close to these values. On the other hand, this constraint makes the system conservative, e.g. in altitude control.

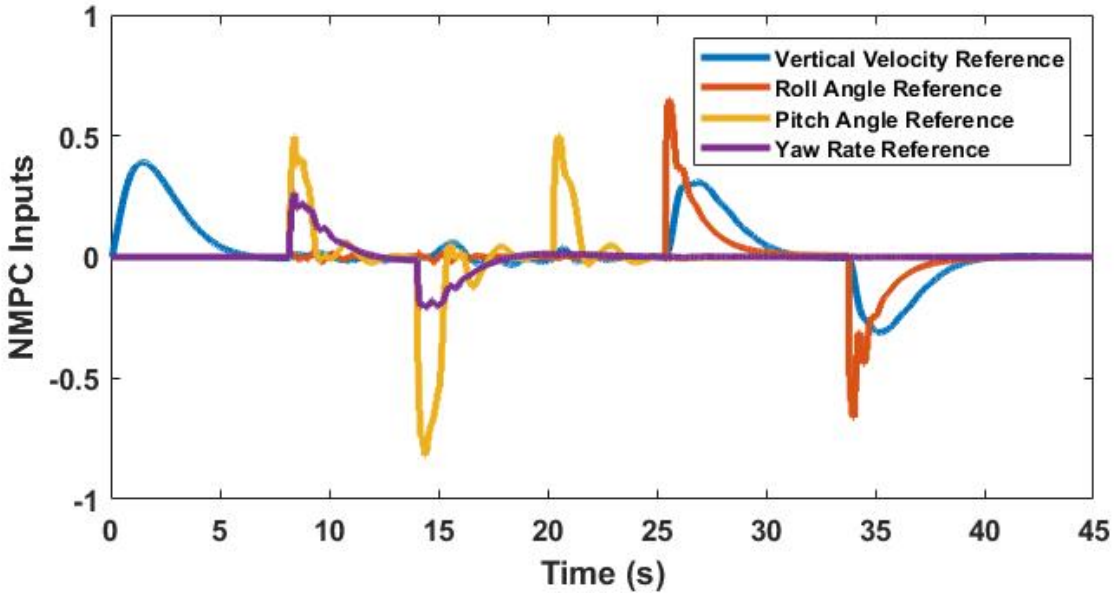

Figure 4.12: NMPC inputs

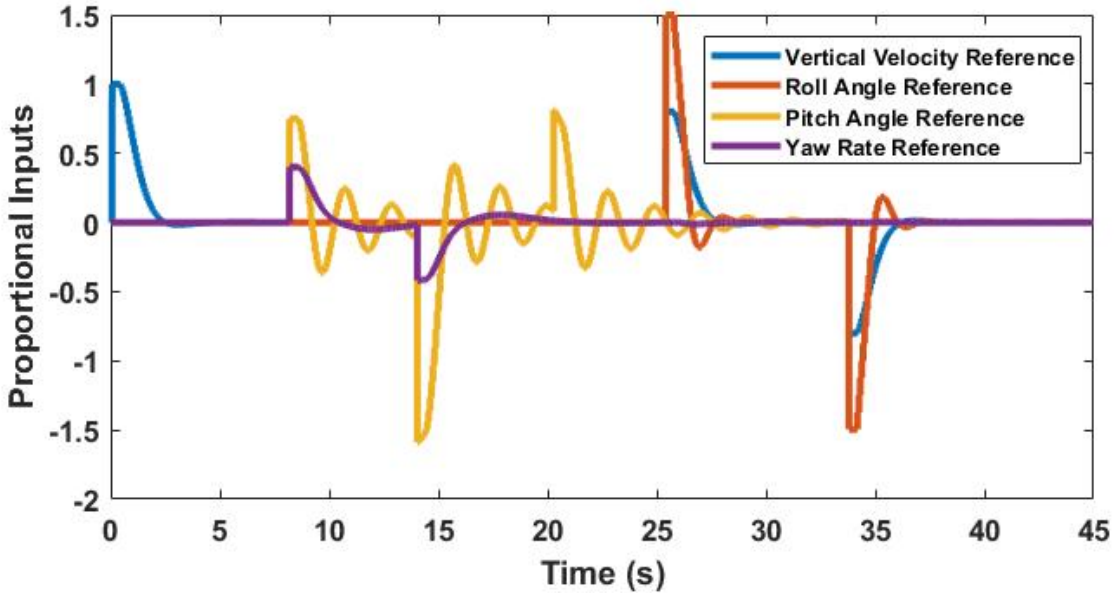

Figure 4.13: Proportional inputs

The NMPC's parameters are tuned by try and error. Figure 4.14 illustrates the effect of N, prediction horizon length, on x controller. By extending the length of the receding horizon, the NMPC predicts the state vector for a longer future time. Therefore, the performance of the controller will improve. However, As shown in Figure 4.14, this improvement is slightly noticeable. Moreover, by increasing the value of N, the complexity of NMPC's optimization problem will grow and the computation of optimum input will be more costly. On the other hand, during tuning, it is noted that for small values of N the system will be highly unstable. Figure 4.15 depicts a scenario in which small N destabilizes x controller. This Figure obtained for $\Delta\tau = 0.3\ s$ and $N = 5$. Therefore, for $\Delta\tau = 0.3$, N should be equal or greater than 6 to avoid instability.

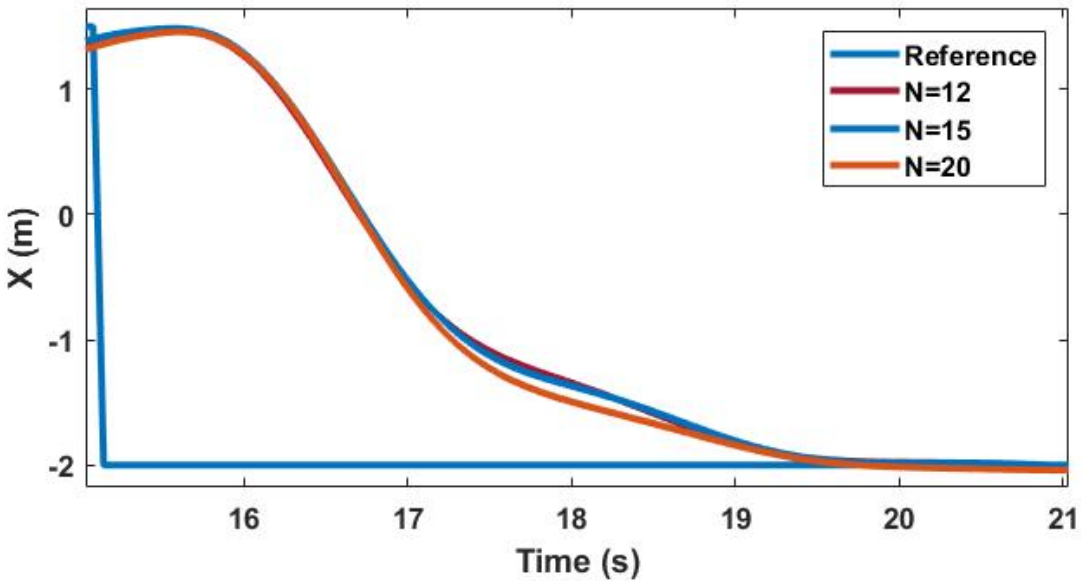

Figure 4.14: Effect of N on x controller

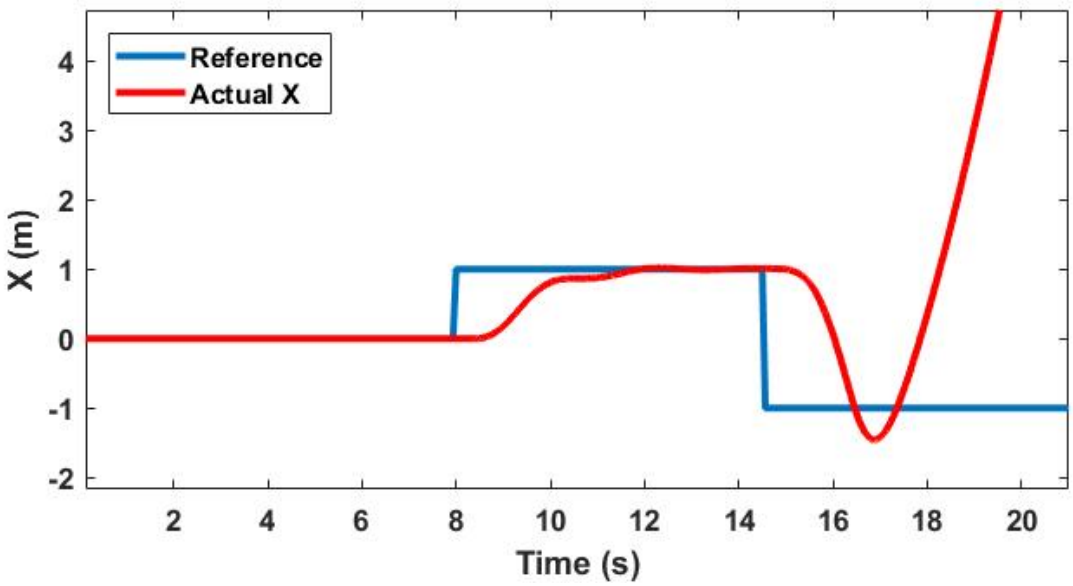

Figure 4.15: Unstable x controller because of small N

Also, Figure 4.16 depicts the effect of $\Delta\tau$ on the performance of x controller. the designed NMPC discrete the continuous control-oriented model by this parameter. By reducing $\Delta\tau$, the discrete model is more realistic. As a result, as Figure 4.16 represents, x controller performs more efficient. However, by decreasing this parameter, the prediction horizon decreases and the system might go to the instability point. Figure 4.17 shows an example of instability because of short value of $\Delta\tau$. For $N = 15$, $\Delta\tau$ should be greater than 0.1 s. Therefore, in Figure 4.17, when $\Delta\tau$ is 0.08 s, the NMPC is highly unstable.

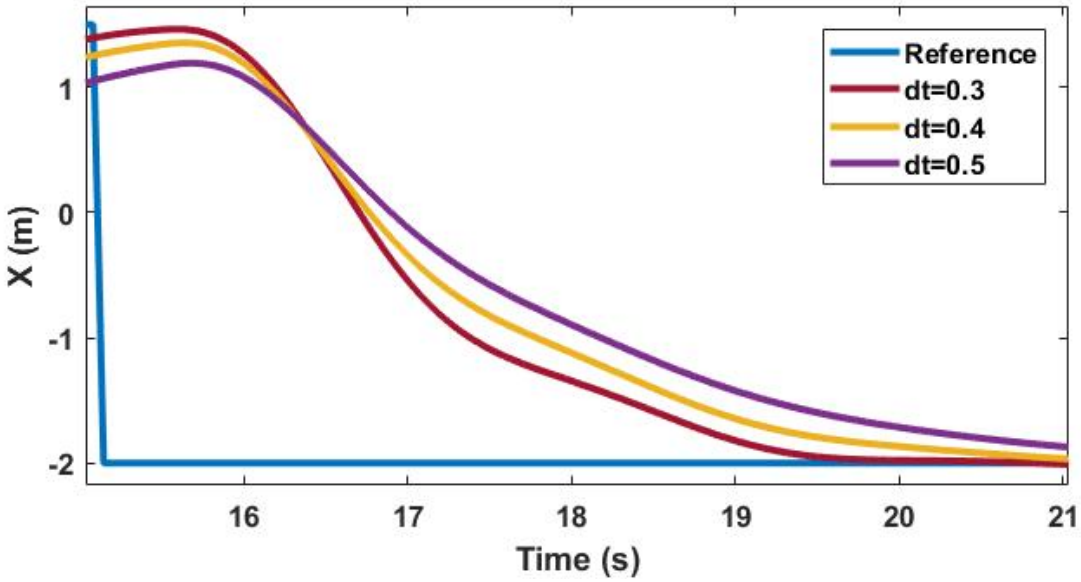

Figure 4.16: Effect of $\Delta\tau$ on x controller

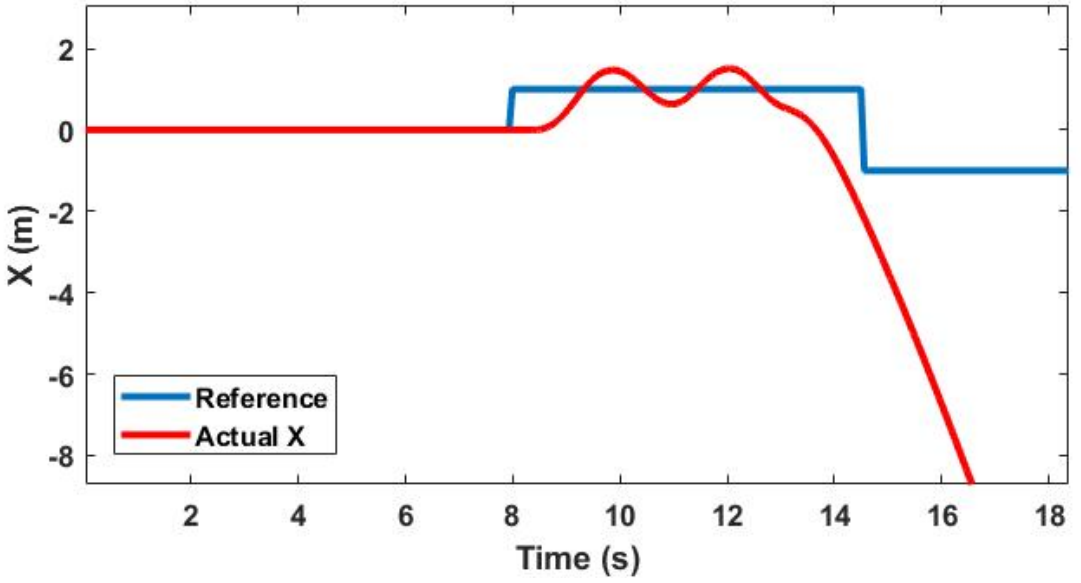

Figure 4.17: Unstable x controller because of small $\Delta\tau$

Finally, Figure 4.18 represents the effect of $\mathbf{q}$ and $\mathbf{r}$ on x controller. Based on equation 3.14, these parameters show the importance of minimizing error or input and are tuned by try and error as diagonal matrices with the elements of vectors $[12, 2, 7, 2, 5, 10, 13]^T$ and $[4, 12, 12, 24]^T$ on the main diagonals. The result using these matrices is shown as Case 2 in Figure 4.18. In this Figure, this graph is compared with two other cases. In these cases, $\mathbf{q}$ and $\mathbf{r}$ generated by applying the following vectors:

- Case 1: $[12, 2, 7, 2, 5, 10, 13]^T \quad [10, 30, 30, 60]^T$
- Case 3: $[12, 2, 7, 2, 5, 10, 13]^T \quad [1, 3, 3, 6]^T$

in contrast to Case 1, Case 3 shows that the value of the input is not as important as minimizing error. Therefore, the controller converges to the desired reference more aggressively with a small overshoot. On the other hand, in Case 1, the graph converges smoother with a tiny more settling time.

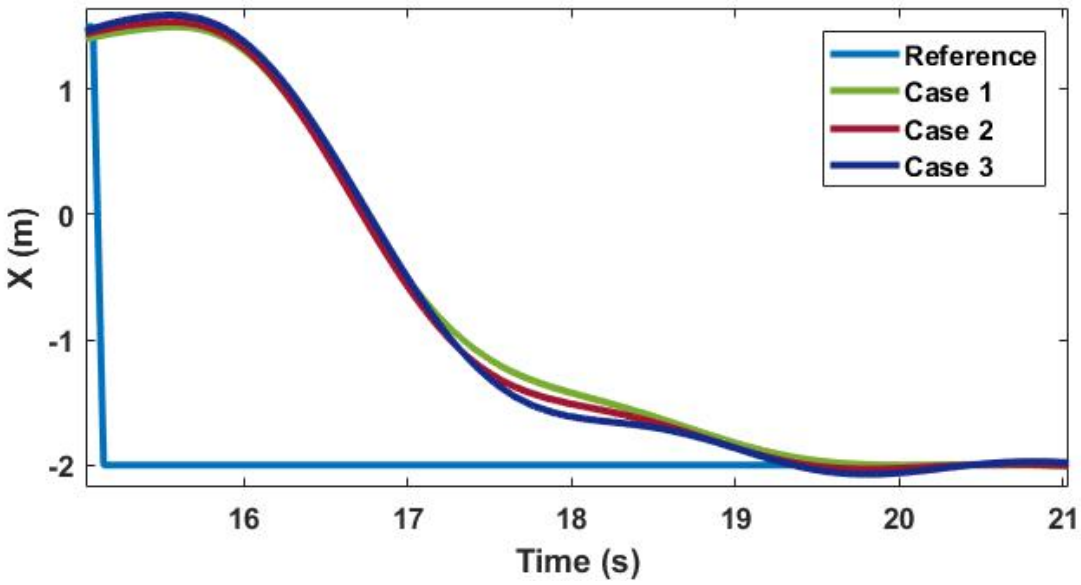

Figure 4.18: Effect of $\mathbf{r}$ on x controller

Also, the time required for NMPC to calculate the optimum input, for a sample has been analyzed. This control approach for this sample needed the average time of 0.12 ms to solve the optimization problem, while the maximum time required was 1.4 ms. This

shows that this approach is considerably fast, and it can surely manage to calculate a proper input at each step-time which is 65 ms.

4.7 Experiments

Figure 4.19 and 4.20 represent a sample of the performance of the estimators. These graphs show that the estimated values can represent the actual vertical velocity with little errors, and in Figure 4.21 and 4.22, it is demonstrated that, even with such errors, the designed NMPC controller can follow the reference height properly. It should be mentioned that, without the Kalman filter, the devised NMPC controller cannot be implemented because the system will be unstable in practice.

Computing the settling time showed that time required for NMPC to take off, reach, and remain to 1 m altitude is 6 s. As discussed, the simulation predicted that this time will be around 5 s. This error might have several reasons; firstly, the simulation model does not represent the actual quadrotor perfectly. Therefore, the experiment of NMPC tuned based on the simulated model will not perform as well as the simulation one. Also, no model can represent an object as well as possible. Therefore, the control-oriented model is incomplete. This model does not consider any kind of uncertainty. Moreover, in this model the value of parameter g is considered as 9.81 $^{m}/_{s^2}$, while it is not the most accurate value all the time. As a result, similar to any other platform, the practical result is not as well as the predicted one.

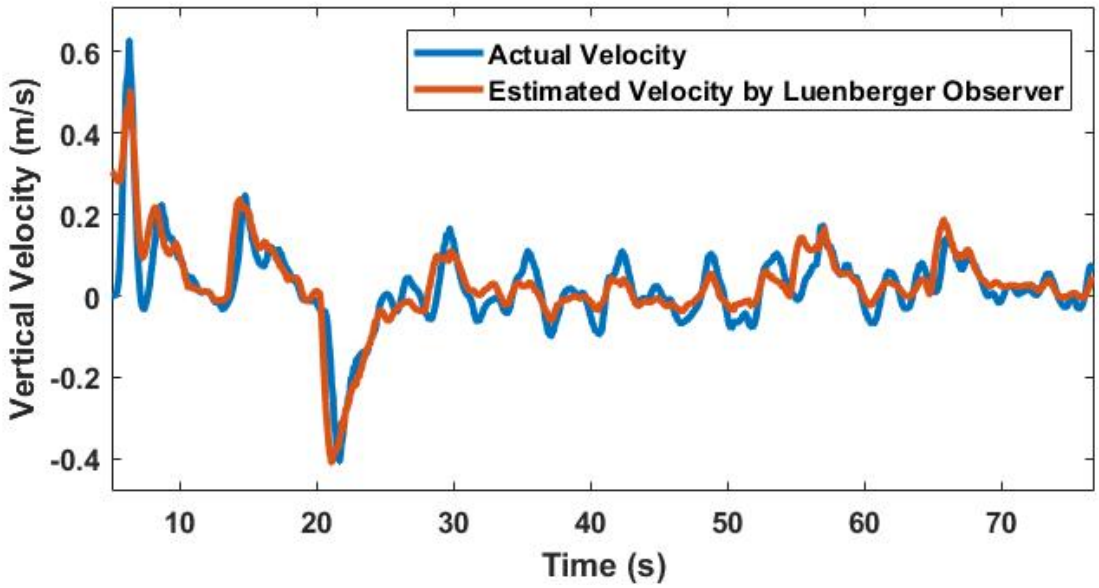

Figure 4.19: A sample of estimated vertical velocity by Kalman filter in practice

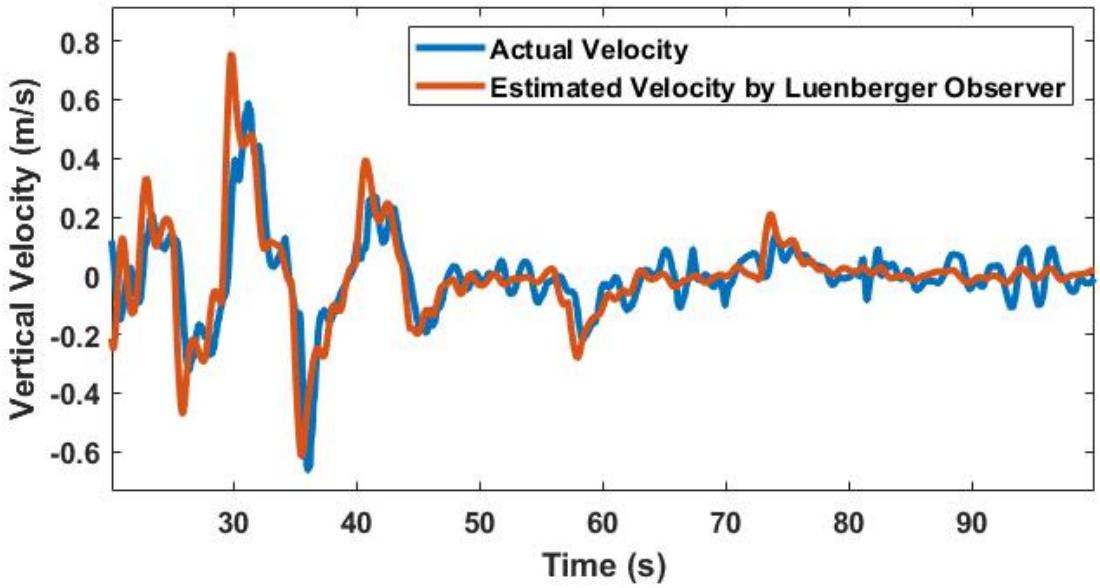

Figure 4.20: A sample of estimated vertical velocity by Luenberger observer in practice

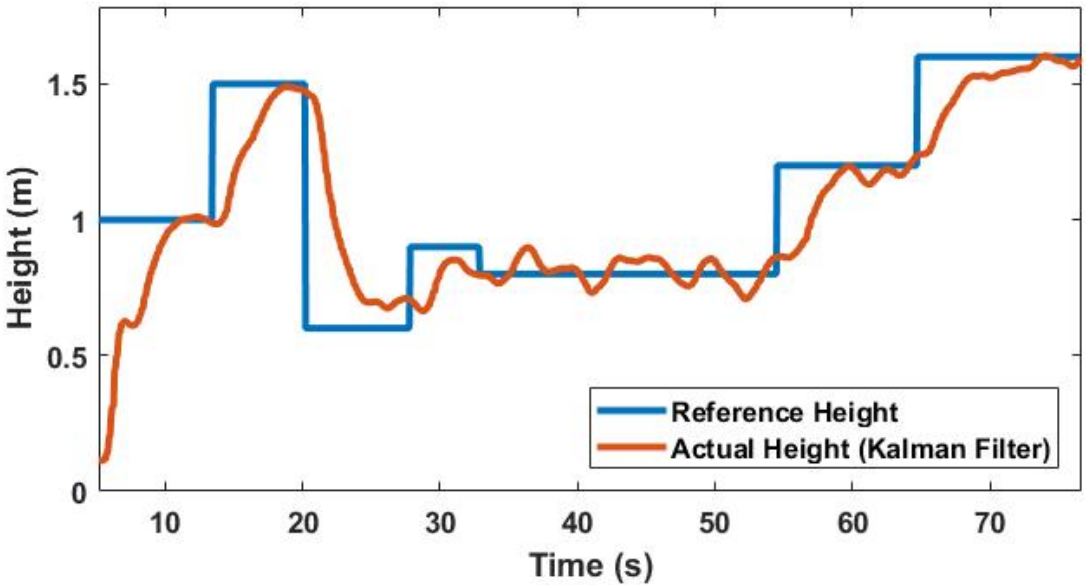

Figure 4.21: Experiment of height control with applying Kalman filter

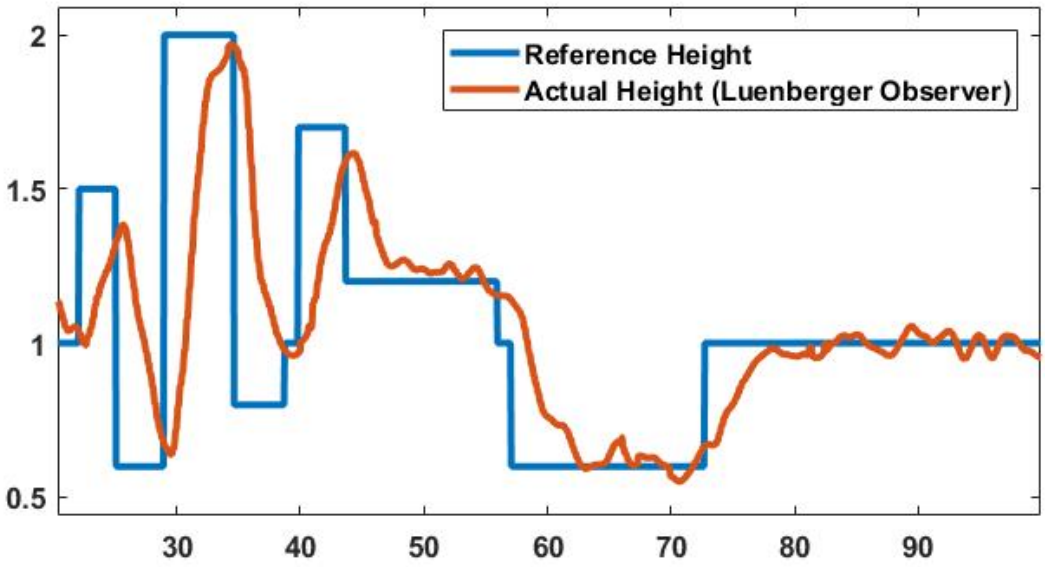

Figure 4.22: Experiment of height control with applying Luenberger observer

Furthermore, in Figure 4.23, the drone is attempting to follow a square path. The blue line is the desired path and the red line is the actual one. As shown, the quadrotor remains close to the desired path with little errors. Moreover, in Figure 4.24, the controller is tested on a more complex path. This path is a square path with different altitude at each corner. The result shows that in contrast to Figure 4.23, errors are increased due to the complexity of manoeuvres, but the quadrotor is still close to the desired path. In this scenario, at some points, the x, y, and z controller apply commands at the same time, while in square path (Figure 4.23) at most two of them perform simultaneously. Therefore, because of the coupled dynamics, the deviation of Figure 4.23 is increased from 0.1045 m to 0.1097 m in Figure 4.24.

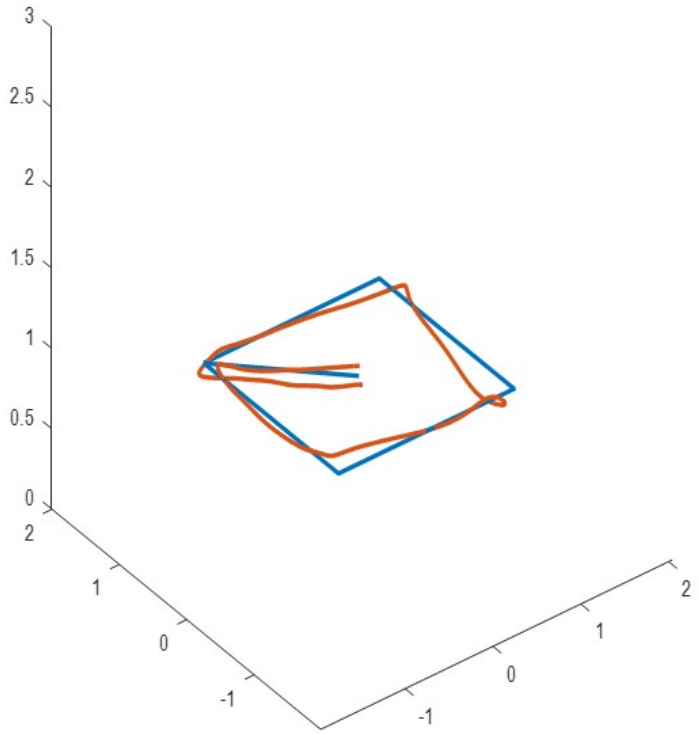

Figure 4.23: Square path following experimental results

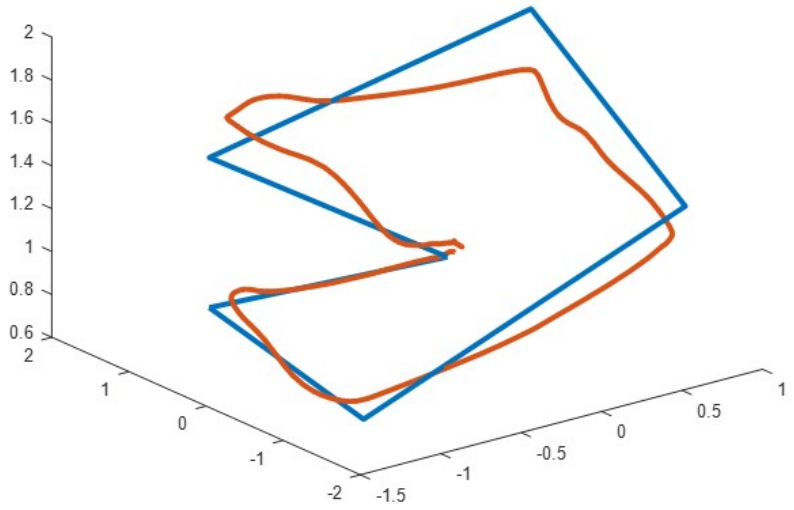

Figure 4.24: Square with different altitude at each corner path following experimental results by applying Kalman filter

4.8 Summary and Remarks

As discussed in chapter 3, $\dot{z}$ is an unknown state which is not measured by onboard and off-board sensors. Therefore, the result of NMPC which is a feedback-based controller is not satisfactory. In this chapter, to overcome this issue, two different state estimation methods were introduced to measure the unknown vertical velocity: the Kalman filter and the Luenberger observer. To make sure these estimation methods are implementable, the observability of the system was checked by deriving the observability matrix. Some practical tests were generated to define the Kalman filter's parameters. Applying these tests led us to define the process and measurement errors covariances. Also, by using some toolboxes, the NMPC was simulated in MATLAB/Simulink and the result showed the effect of the designed estimators. Besides, the comparison between the Kalman filter and the Luenberger observer showed similar improvements. Furthermore, the NMPC was compared with a proportional controller, and the result showed the superior performance of NMPC in contrast to the proportional scheme in most cases. Moreover, these approaches are tested on AR.Drone 2.0 which is a commercial drone, while Vicon Vantage motion

capture system was used as an off-board measurement device. The result showed that this control approach can keep the drone on even complex paths, i.e. square path with a small error.

Chapter 5

Concluding Remarks

5.1 Summary of Contributions

This study can be summarized as follows:

- To solve the trajectory tracking problem for quadrotors a fast-implementable NMPC-based controller was designed to perform as a high-level control module.
  - The control-oriented model was derived. This model was highly nonlinear but more realistic.
  - The order of non-linearity made the NMPC's real-time optimizations very time-consuming for practical applications. To resolve this issue, the controller was embedded with a fast optimizer, namely the Newton/GMRES scheme. This method solves the optimization problem by applying the Hamiltonian and fdgmres methods.
  - This control approach is simulated and experimented utilizing a commercial drone, AR.Drone 2.0, and a set of off-board cameras, Vicon Vantage motion capture system. To implement this method, ARDrone Simulink Development Kit V1.1 [55] and MPSee [60] Simulink toolboxes were used.
  - The defective result of the simulated NMPC showed that the performance of NMPC depends on the accuracy of feedback. Also, due to the instability of the system, this approach could experiment successfully.

- NMPC can be categorized as a feedback-based controller. Therefore, to provide the unknown vertical velocity, some state estimation methods were discussed.
  - To check the observability of the system which ensures that state estimation techniques are implementable, the observability matrix is stated and checked to be a full rank matrix.
  - A Kalman filter and Luenberger observer were designed and employed to determine the unknown vertical velocity for real-life implementations. Kalman filter's parameters, such as the process and measurement covariances, were determined by analyzing some practical data.
  - The result of simulation showed the great effect of estimation tools on the performance of NMPC. Also, the comparison between these two observers showed similar improvements.
  - The proposed NMPC control system was compared with a proportional controller in simulations, which indicated significant improvements in the control performance.
  - This technique was successfully implemented on AR.Drone 2.0 using the Vicon Vantage motion capture system to track the drone. The test results demonstrated the superior performance of the proposed NMPC-based controller.

5.2 Future Work

Some potential future work to expand this study is as follows:

- Examine whether the controller can follow other complicated trajectories, such as Elliptical and Lorenz paths.
- Solving the obstacle avoidance problem using the proposed control scheme. It can be done by adding an inequality constraint in NMPC problem definition, such as $(P_q - P_o)^2 > d^2$, where P_q and P_o are the position of quadcopter and obstacle and d is the desired distance. This inequality constraint keeps the quadrotor away from pre-defined obstacles.
- The performance of the NMPC-based controller can be compared with another advanced method like Reinforcement Learning (RL).

- Following dynamic objects like scale cars in scale smart city by quadrotors.
- The estimation approaches can be applied to any dynamic objects working under the vision of the Vicon motion capture system.
- Diagnosing sensors or actuators fault as by adding uncertainty terms in the control-oriented model.
- Considering inherent delay and bandwidth of actuators in the model development for controller design

References

[1] Mahyar Abdolhosseini, YM Zhang, and Camille Alain Rabbath. An efficient model predictive control scheme for an unmanned quadrotor helicopter. *Journal of intelligent & robotic systems*, 70(1-4):27–38, 2013.

[2] Kostas Alexis, George Nikolakopoulos, and Anthony Tzes. Model predictive quadrotor control: attitude, altitude and position experimental studies. *IET Control Theory & Applications*, 6(12):1812–1827, 2012.

[3] Kostas Alexis, George Nikolakopoulos, and Anthony Tzes. On trajectory tracking model predictive control of an unmanned quadrotor helicopter subject to aerodynamic disturbances. *Asian Journal of Control*, 16(1):209–224, 2014.

[4] Sina Alighanbari and Nasser L Azad. Ecological nmpc controller for connected and automated plug-in hybrid electric vehicles at roundabouts. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1069–1074. IEEE, 2019.

[5] Frank Allgöwer and Alex Zheng. *Nonlinear model predictive control*, volume 26. Birkhäuser, 2012.

[6] Erdinc Altug, James P Ostrowski, and Robert Mahony. Control of a quadrotor helicopter using visual feedback. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 1, pages 72–77. IEEE, 2002.

[7] Yann Ameho, Fabien Niel, François Defaÿ, Jean-Marc Biannic, and Caroline Bérard. Adaptive control for quadrotors. In *2013 IEEE International Conference on Robotics and Automation*, pages 5396–5401. IEEE, 2013.

[8] Anil Aswani, Patrick Bouffard, and Claire Tomlin. Extensions of learning-based model predictive control for real-time application to a quadrotor helicopter. In *2012 American Control Conference (ACC)*, pages 4661–4666. IEEE, 2012.

[9] Anil Aswani, Humberto Gonzalez, S Shankar Sastry, and Claire Tomlin. Provably safe and robust learning-based model predictive control. *Automatica*, 49(5):1216–1226, 2013.

[10] Craig Earl Beal and J Christian Gerdes. Model predictive control for vehicle stabilization at the limits of handling. *IEEE Transactions on Control Systems Technology*, 21(4):1258–1269, 2012.

[11] Michael Bloesch, Marco Hutter, Mark A Hoepflinger, Stefan Leutenegger, Christian Gehring, C David Remy, and Roland Siegwart. State estimation for legged robots-consistent fusion of leg kinematics and imu. *Robotics*, 17:17–24, 2013.

[12] Hoseinali Borhan, Ardalan Vahidi, Anthony M Phillips, Ming L Kuang, Ilya V Kolmanovsky, and Stefano Di Cairano. Mpc-based energy management of a power-split hybrid electric vehicle. *IEEE Transactions on Control Systems Technology*, 20(3):593–603, 2011.

[13] Samir Bouabdallah, Andre Noth, and Roland Siegwart. Pid vs lq control techniques applied to an indoor micro quadrotor. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2451–2456. IEEE, 2004.

[14] Hakim Bouadi, M Bouchoucha, and M Tadjine. Sliding mode control based on backstepping approach for an uav type-quadrotor. *World Academy of Science, Engineering and Technology*, 26(5):22–27, 2007.

[15] Patrick Bouffard, Anil Aswani, and Claire Tomlin. Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results. In *2012 IEEE International Conference on Robotics and Automation*, pages 279–284. IEEE, 2012.

[16] Kemal Buyukkabasakal, Baris Fidan, and Aydogan Savran. Mixing adaptive fault tolerant control of quadrotor uav. *Asian Journal of Control*, 19(4):1441–1454, 2017.

[17] David Cabecinhas, Rita Cunha, and Carlos Silvestre. A globally stabilizing path following controller for rotorcraft with wind disturbance rejection. *IEEE Transactions on Control Systems Technology*, 23(2):708–714, 2014.

[18] Gang Cao, Edmund M-K Lai, and Fakhrul Alam. Gaussian process model predictive control of an unmanned quadrotor. *Journal of Intelligent & Robotic Systems*, 88(1):147–162, 2017.

[19] P Castillo, R Lozano, and A Dzul. Stabilization of a mini rotorcraft with four rotors. ieee control systems magazine (2005), 45-55. l. 2005.

[20] Pedro Castillo, Alejandro Dzul, and Rogelio Lozano. Real-time stabilization and tracking of a four-rotor mini rotorcraft. *IEEE Transactions on control systems technology*, 12(4):510–516, 2004.

[21] Zhou Chao, Lei Ming, Zhou Shaolei, and Zhang Wenguang. Collision-free uav formation flight control based on nonlinear mpc. In *2011 international conference on electronics, communications and control (ICECC)*, pages 1951–1956. IEEE, 2011.

[22] Ming Chen and Mihai Huzmezan. A combined mbpc/2 dof h infinity controller for a quad rotor uav. In *AIAA guidance, navigation, and control conference and exhibit*, page 5520, 2003.

[23] Jan Dentler, Somasundar Kannan, Miguel Angel Olivares Mendez, and Holger Voos. A real-time model predictive position control with collision avoidance for commercial low-cost quadrotors. In *2016 IEEE conference on control applications (CCA)*, pages 519–525. IEEE, 2016.

[24] Zachary Dydek, Anuradha Annaswamy, and Eugene Lavretsky. Combined/composite adaptive control of a quadrotor uav in the presence of actuator uncertainty. In *AIAA Guidance, Navigation, and Control Conference*, page 7575, 2010.

[25] Zachary T Dydek, Anuradha M Annaswamy, and Eugene Lavretsky. Adaptive control of quadrotor uavs: A design trade study with flight evaluations. *IEEE Transactions on control systems technology*, 21(4):1400–1406, 2012.

[26] Lars Grüne and Jürgen Pannek. Nonlinear model predictive control. In *Nonlinear Model Predictive Control*, pages 45–69. Springer, 2017.

[27] Chingiz Hajiyev and Halil Ersin Soken. Robust adaptive kalman filter for estimation of uav dynamics in the presence of sensor/actuator faults. *Aerospace Science and Technology*, 28(1):376–383, 2013.

[28] Gabriel Hoffmann, Steven Waslander, and Claire Tomlin. Distributed cooperative search using information-theoretic costs for particle filters, with quadrotor applications. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, page 6576, 2006.

[29] Mike Huang, Hayato Nakada, Ken Butts, and Ilya Kolmanovsky. Nonlinear model predictive control of a diesel engine air path: A comparison of constraint handling and computational strategies. *IFAC-PapersOnLine*, 48(23):372–379, 2015.

[30] Petros Ioannou and Baris Fidan. *Adaptive control tutorial*. SIAM, 2006.

[31] Mohamad Iskandarani, Sidney N Givigi, Camille Alain Rabbath, and Alain Beaulieu. Linear model predictive control for the encirclement of a target using a quadrotor aircraft. In *21st Mediterranean Conference on Control and Automation*, pages 1550–1556. IEEE, 2013.

[32] S Islam, PX Liu, and A El Saddik. Nonlinear adaptive control for quadrotor flying vehicle. *Nonlinear Dynamics*, 78(1):117–133, 2014.

[33] Shafiqul Islam, Peter X Liu, and Abdulmotaleb El Saddik. Robust control of four-rotor unmanned aerial vehicle with disturbance uncertainty. *IEEE Transactions on Industrial Electronics*, 62(3):1563–1571, 2014.

[34] Yeonsik Kang and J Hedrick. Design of nonlinear model predictive controller for a small fixed-wing unmanned aerial vehicle. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, page 6685, 2006.

[35] Carl T Kelley. *Iterative methods for linear and nonlinear equations*, volume 16. Siam, 1995.

[36] Hossein Bonyan Khamseh and Farrokh Janabi-Sharifi. Ukf–based lqr control of a manipulating unmanned aerial vehicle. *Unmanned Systems*, 5(03):131–139, 2017.

[37] Donald E Kirk. *Optimal control theory: an introduction*. Courier Corporation, 2004.

[38] Nasrettin Koksal. *Adaptive and Optimal Motion Control of Multi-UAV Systems*. PhD thesis, University of Waterloo, 2019.

[39] Nasrettin Koksal, Baris Fidan, and Kemal Buyukkabasakal. Real-time implementation of decentralized adaptive formation control on multi-quadrotor systems. In *2015 European Control Conference (ECC)*, pages 3162–3167. IEEE, 2015.

[40] Benoit Landry et al. *Planning and control for quadrotor flight through cluttered environments*. PhD thesis, Massachusetts Institute of Technology, 2015.

[41] Robert C Leishman, John C Macdonald, Randal W Beard, and Timothy W McLain. Quadrotors and accelerometers: State estimation with an improved dynamic model. *IEEE Control Systems Magazine*, 34(1):28–41, 2014.

[42] Mostafa Mohammadi and Alireza Mohammad Shahri. Adaptive nonlinear stabilization control for a quadrotor uav: Theory, simulation and experimentation. *Journal of Intelligent & Robotic Systems*, 72(1):105–122, 2013.

[43] Mohd Ariffanan Mohd Basri, Abdul Rashid Husain, and Kumeresan A Danapalasingam. Intelligent adaptive backstepping control for mimo uncertain non-linear quadrotor helicopter systems. *Transactions of the Institute of Measurement and Control*, 37(3):345–361, 2015.

[44] Kartik Mohta. *State Estimation, Control, and Planning for a Quadrotor Team*. PhD thesis, Publicly Accessible Penn Dissertations, 2018.

[45] D Subbaram Naidu. *Optimal control systems*. CRC press, 2002.

[46] C Nicol, CJB Macnab, and A Ramirez-Serrano. Robust adaptive control of a quadrotor helicopter. *Mechatronics*, 21(6):927–938, 2011.

[47] Toshiyuki Ohtsuka. A continuation/gmres method for fast computation of nonlinear receding horizon control. *Automatica*, 40(4):563–574, 2004.

[48] Christos Panos, Konstantinos Kouramas, and Efstratios N Pistikopoulos. Robust explicit/multi—parametric model predictive control for box-constrained linear dynamic systems. *IFAC Proceedings Volumes*, 43(5):212–217, 2010.

[49] Stephane Piskorski, Nicolas Brulez, Pierre Eline, and Frederic D'Haeyer. Ar. drone developer guide. *Parrot, sdk*, 1, 2012.

[50] Johann-Sebastian Pleban, Ricardo Band, and Reiner Creutzburg. Hacking and securing the ar. drone 2.0 quadcopter: investigations for improving the security of a toy. In *Mobile Devices and Multimedia: Enabling Technologies, Algorithms, and Applications 2014*, volume 9030, page 90300L. International Society for Optics and Photonics, 2014.

[51] Guilherme V Raffo, Manuel G Ortega, and Francisco R Rubio. An integral predictive/nonlinear H_∞ control structure for a quadrotor helicopter. *Automatica*, 46(1):29–39, 2010.

[52] Gerasimos G Rigatos. Nonlinear kalman filters and particle filters for integrated navigation of unmanned aerial vehicles. *Robotics and Autonomous Systems*, 60(7):978–995, 2012.

[53] Youcef Saad and Martin H Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986.

[54] M Foad Samadi and Mehrdad Saif. Nonlinear model predictive control for cell balancing in li-ion battery packs. In *2014 American Control Conference*, pages 2924–2929. IEEE, 2014.

[55] DE Sanabria and P Mosterman. Ardrone simulink development kit v1. 1. http://www.mathworks.com/matlabcentral/fileexchange/43719-ar-drone-simulink-development-kit-v1-1/, 2013.

[56] Lucas Vago Santana, Alexandre Santos Brandao, and Mario Sarcinelli-Filho. Outdoor waypoint navigation with the ar. drone quadrotor. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 303–311. IEEE, 2015.

[57] Mohammad Sarim, Alireza Nemati, Manish Kumar, and Kelly Cohen. Extended kalman filter based quadrotor state estimation based on asynchronous multisensor data. In *ASME 2015 Dynamic Systems and Control Conference*. American Society of Mechanical Engineers Digital Collection, 2015.

[58] Justin M Selfridge and Gang Tao. A multivariable adaptive controller for a quadrotor with guaranteed matching conditions. *Systems Science & Control Engineering: An Open Access Journal*, 2(1):24–33, 2014.

[59] Payman Shakouri, Andrzej Ordys, and Mohamad R Askari. Adaptive cruise control with stop&go function using the state-dependent nonlinear model predictive control approach. *ISA transactions*, 51(5):622–631, 2012.

[60] Sadegh Tajeddin. Automatic code generation of real-time nonlinear model predictive control for plug-in hybrid electric vehicle intelligent cruise controllers. Master's thesis, University of Waterloo, 2016.

[61] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

[62] Holger Voos. Nonlinear control of a quadrotor micro-uav using feedback-linearization. In *2009 IEEE International Conference on Mechatronics*, pages 1–6. IEEE, 2009.

[63] Dong Wang, Quan Pan, Jinwen Hu, Chunhui Zhao, and Yaning Guo. Mpcc-based path following control for a quadrotor with collision avoidance guaranteed in constrained environments. In *2019 IEEE 28th International Symposium on Industrial Electronics (ISIE)*, pages 581–586. IEEE, 2019.

[64] Honglin Wang and Mou Chen. Trajectory tracking control for an indoor quadrotor uav based on the disturbance observer. *Transactions of the Institute of Measurement and Control*, 38(6):675–692, 2016.

[65] Lu Wang and Jianbo Su. Robust disturbance rejection control for attitude tracking of an aircraft. *IEEE Transactions on Control Systems Technology*, 23(6):2361–2368, 2015.

[66] Jingyun Wu, Guoliang Liu, and Tao Huang. Noise covariance identification using autocovariance least-squares technique for state estimation of quadrotor. In *2017 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, pages 88–93. IEEE, 2017.

[67] Ted Xiao. A literature review of learning and optimization methods applied to quadrotor control. 2016.

[68] Jing-Jing Xiong and En-Hui Zheng. Optimal kalman filter for state estimation of a quadrotor uav. *Optik*, 126(21):2862–2868, 2015.

[69] Kwangjin Yang, Yeonsik Kang, and Salah Sukkarieh. Adaptive nonlinear model predictive path-following control for a fixed-wing unmanned aerial vehicle. *International Journal of Control, Automation and Systems*, 11(1):65–74, 2013.

[70] Kaijiang Yu, Masakazu Mukai, and Taketoshi Kawabe. A battery management system using nonlinear model predictive control for a hybrid electric vehicle. *IFAC Proceedings Volumes*, 46(21):301–306, 2013.

[71] Andrea Zanelli, Greg Horn, Gianluca Frison, and Moritz Diehl. Nonlinear model predictive control of a human-sized quadrotor. In *2018 European Control Conference (ECC)*, pages 1542–1547. IEEE, 2018.

[72] Bo Zhao, Bin Xian, Yao Zhang, and Xu Zhang. Nonlinear robust sliding mode control of a quadrotor unmanned aerial vehicle based on immersion and invariance method. *International Journal of Robust and Nonlinear Control*, 25(18):3714–3731, 2015.

[73] Zongyu Zuo and Chenliang Wang. Adaptive trajectory tracking control of output constrained multi-rotors systems. *IET Control Theory & Applications*, 8(13):1163–1174, 2014.