Adaptive Model Predictive Control for a Delivery Quadrotor

Author:
Piwai Nigel CHIKASHA

Supervisor:
Chioniso KUCHWA-DUBE

A Dissertation submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in partial fulfilment of the requirements of the degree of Master of Science.

November 23, 2018

Declaration of Authorship

I, Piwai Nigel CHIKASHA, declare that this dissertation titled, "Adaptive Model Predictive Control for a Delivery Quadrotor" is entirely my own work except where otherwise stated. I confirm that this research work was done while in candidature at the University of the Witwatersrand and no part of this research dissertation has previously been submitted for any other qualification at any other institution. Where I have consulted the published work of others, this is always clearly attributed and I have acknowledged all main references. I also understand what plagiarism is and I am aware of the University's policies in this regard.

Signed:

_____

"Go down deep enough into anything and you will find mathematics."

Dean Schlicter

Abstract

This work explores the application of Adaptive Model Predictive Control (AMPC) to a quadrotor, given some delivery (varying load) task. In simulations, this is investigated as a pick-and-place task for a delivery quadrotor, then in experiments, as a load-drop-off stabilisation task. Model Predictive Control (MPC) is a very powerful method of Advanced Control, utilizing an implicit model to make output predictions which are in turn used in computing control action. Dynamic Matrix Control (DMC) is very common in process control today, especially for chemical plants. Due to changes in operating points, the implicit model in MPC typically becomes insufficient, costing the quality of the controller. This work proposes Adaptation (based on the Recursive Least Squares (RLS) algorithm), for online system identification to update the model in real time hence taking changes in operating points into account when computing the DMC control action. Overall, Adaptive DMC (ADMC) is investigated as to whether ADMC is capable of improved quadrotor control given load variation. The quadrotor is modelled analytically based on the Newton Euler Formalisation then experimentally based on system identification from flight data. The experimental model is optimised before being adopted as the system model for ADMC design. The adaptive controller is implemented and simulations run to evaluate the controller against PID and also adaptive PID. The controller comparison is based on experimental graphical analysis, which is possible if data is well presented. Optimal controller tuning methods are also a common basis for controller evaluation, producing precise evaluation indices for controller comparison, although typically applied to theoretical cases. The controllers are also tested experimentally, on a quadrotor and evaluated, for different load disturbance magnitudes. The experiments reveal that ADMC provides the best control quality in terms of disturbance compensation. To achieve this though, requires careful tuning of the possible ADMC parameters and in this case, additional manipulation of control action by applying some signal smoothing algorithm to the control action. The comparison is made by analysing, how in each case, the quadrotor responds to an in-flight weight (load) drop-off. These experiments are also important in understanding how close to reality system simulation is, and if not close enough, then the reasons for this too. An important aspect that is kept in mind in the design process is that of minimising computation because the ADMC controller (which must solve an optimisation problem at each time sample) is computationally expensive as it is.

Acknowledgements

I would like to thank my supervisor Ms Chioniso Kuchwa-Dube for her support and assistance in this work. I am grateful, for her support in mobilising the equipment, test space and training required for the success of this work. Special thanks also, for her experimental supervision, making sure that experiments were conducted in a safe manner and all guiding protocols observed as required. I would like to thank my family for their encouragement, moral support and financial support. My parents Jeniffar and Adolph Chikasha had a huge role to play and I am grateful.

Publication statement

The following publication resulted from this dissertation:

P. Chikasha and C. Kuchwa-Dube, "Adaptive Model Predictive Control of a quadrotor," Control Conference Africa (CCA), South Africa, 2017, Vol. 50, Issue 2, pp. 157 – 162.

Contents

List of Figures

List of Tables

List of Abbreviations

ADMC	**A**daptive **D**ynamic **M**atrix **C**ontrol
AMPC	**A**daptive **M**odel **P**redictive **C**ontrol
ARX	**A**uto **R**egressive **E**xogenouos
DMC	**D**ynamic **M**atrix **C**ontrol
GPS	**G**eneralised **P**redictive **C**ontrol
GPS	**G**lobal **P**ositioning **S**ystem
IAE	**I**ntegral **A**bsolute **E**rror
IMU	**I**nertial **M**easurement **U**nit
ITSE	**I**ntegral of **T**ime Multiply **S**quared **E**rror
LQR	**L**inear **Q**uadratic **R**egulator
LTV	**L**inear **T**ime **V**arying
MIMO	**M**ultiple **I**nput **M**ultiple **O**utput
MPC	**M**odel **P**redictive **C**ontrol
PSP	**P**ilot **S**upport **P**ackage
PID	**P**roportional **I**ntegral **D**erivative
PWM	**P**ulse **W**idth **M**odulated
RLSE	**R**ecursive **L**east **S**quares **E**stimator
RPE	**R**ecursive **P**olynomial **E**stimator
SISO	**S**ingle **I**nput **S**ingle **O**utput
UAV	**U**manned **A**erial **V**ehicle

To my mother Jeniffar and father Adolph . . .

Chapter 1

INTRODUCTION

1.1 Background

An Unmanned Aerial Vehicle (UAV) is an aircraft controlled remotely, without a human pilot onboard. UAVs have a payload capability which means that UAVs can be used as platforms on which objects can be lifted. This capability introduces opportunity for delivery of goods using UAVs over and above the capability to lift useful data acquisition modules such as sensors. UAVs (as platforms on which various sensors can be mounted) are fast becoming important assets in key sectors of economies such as agriculture, mining, mapping, environment management etc. Mounting a magnetometer allows us to execute mineral exploration, while a multi-spectral camera enables agricultural research (crop health assessment, harvest estimation etc). Alternatively, manned aircraft or Satellite Imagery can be used for these vital activities, but not without compromise to accuracy and at a cost. Figure 1.1, has been put together to outline the comparison of these three.

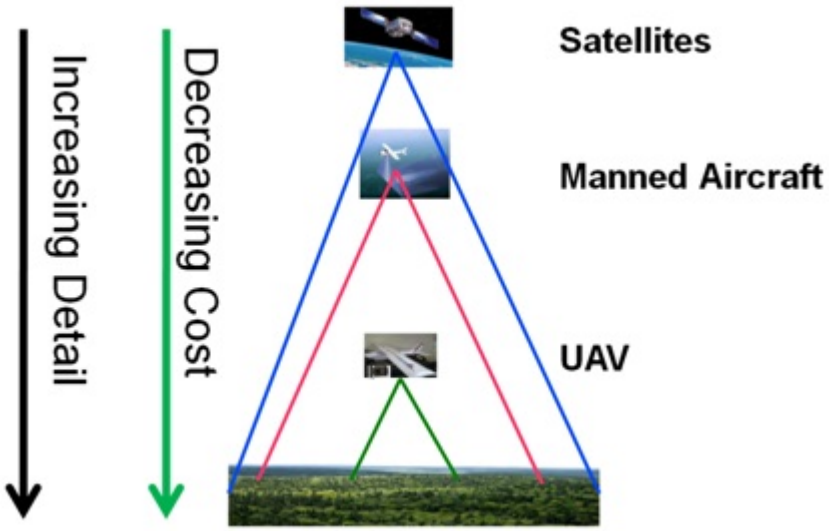

FIGURE 1.1: Comparison of UAV, manned aircraft and satellite imagery

1.2 Purpose of the study

The purpose of the study is to develop a controller based on adaptive model predictive laws, for tests with a quadrotor Unmanned Aerial Vehicle (UAV). This controller is then evaluated against Proportional Integral Differential (PID) and also adaptive PID control. In simulation, the tests investigate the performance of the controllers for a load pick-and-place task with a delivery quadrotor, while in practical experiments, tests are carried out for the varying-load task, where is load is dropped mid-flight. Aspects surrounding the tuning and implementation of the adaptive predictive controller are discussed. Overall, the study establishes whether or not, the adaptive predictive controller is capable of improved control quality.

1.3 Research background and context

The ability of quadrotors to lift loads has shown great potential in application to transportation and aerial manipulation. To this end, some research has been directed to parcel delivery using quadrotors [1], as a possible solution to the difficulty of delivery in remote areas. When the quadrotor is programmed to automatically approach an object to be transported, pick up the object and move the object to some desired point, this is commonly called a 'pick-and-place' task, widely referred to as 'aerial manipulation'. Aerial manipulation also includes cases where it is necessary to interact with the environment physically while airborne, which applies for example, to infrastructural inspection and/or repair. For dangerous operations, using UAVs for this task provides a safer alternative rather than deploying a person to the task. This again, usually involves the pick-and-place task (using some actuated manipulator), if the quadrotor has to pickup a certain component for example a sensor, and place the sensor at some point that would be difficult for a human being to reach.

Transporting an item with a quadrotor or attaching and/or operating a manipulator means adding load to the quadrotor. This will affect the dynamics of the quadrotor. It is therefore necessary to develop controllers that take the load effects into account, or else the system is likely to perform unsatisfactorily in tracking the planned route. Efficient and precise tracking will therefore demand high levels of reliability and quality from the UAV controller. Error in trajectory would likely result in an air crash or loss of the vehicle and the precious load on board. Many different controllers have been tested for the control of quadrotors, from both classical and modern strategies. Whether or not a controller works desirably for quadrotor control normally depends on the model of the quadrotor [2] and the environment the quadrotor will operate in.

1.4 Research motivation

Quadrotors can land or take-off vertically and can hover. This makes the quadrotor a highly recommendable robot for aerial manipulation (in this case delivery). Successful aerial manipulation with a quadrotor will certainly require reliable and efficient control. This research proposes adaptive model predictive control (AMPC) for the quadrotor pick-and-place task. Adaptive control is a learning-based control technique which allows a system to adjust control by automatically updating the system model in real time according to the respective process dynamics. Adaptive control has gained much ground in flight control systems [3, 4] and has become a topic of interest in cases where plant dynamics or disturbances are unknown or varying, as the case with the pick-and-place quadrotor. Adaptive control in this case, will therefore offer a controller with adjustable parameters and a mechanism to adjust the parameters [5, 6] in response to variations in the dynamics of the quadrotor according to, not only the loads picked up, but external or internal disturbances too.

Model predictive control (MPC) is an optimal control technique employing an internal model to predict system behaviour in some finite time horizon. Using MPC, it is possible to achieve:

- Optimal tracking (with compensation of model uncertainty) of the desired trajectory even with the load attached.
- Direct incorporation of constraints to represent physical limitations of the quadrotor such as speed.
- The ability to explicitly handle the multi-variable quadrotor system which has coupled inputs and outputs.

The combination of adaptive control and MPC provides a flexible approach to control design. While there may be standard design methods for adaptive and model predictive control, implementation and tuning of the controllers are entirely open to be tailored to meet the required demands [7].

1.5 Problem statement

The quadrotor must follow a specified trajectory in picking up, transporting and placing a load at some point. It is required that the quadrotor follows this trajectory as closely as possible regardless of the disturbances encountered. The load, with unknown mass, introduces load dynamics to the system according to the mass, size and position of the load. Moreover, the quadrotor also operates subject to unknown wind disturbances and possibly with uncertain dynamics influenced by misalignment of motors, weight imbalance etc. The problem is that, accurate control and tracking of the trajectory is difficult with the stated disturbances acting against the vehicle.

1.6 Research Question

This research proposes AMPC for a pick-and-place quadrotor UAV. The adaptive component must ensure that the system adjusts to the attached load while the MPC component introduces optimal tracking. The research question is whether or not adaptive model predictive control (in the context of Adaptive Dynamic Matrix Control - ADMC) will provide improved control in disturbance rejection and reference tracking in executing a pick-and-place mission using a quadrotor. To answer this question, simulations are run and experiments carried out. In these experiments, different controllers are compared based on a particular quadrotor. The experiments analyse the behaviour of each controller when some identical load is dropped off the quadrotor in-flight. Particularly, this analysis explores deviation from setpoint and ho long it takes for the system to self-correct, and also the steady-state error.

1.7 Research Objectives

The objectives of the study are to:

- Review relevant critical literature.
- Model the quadrotor system analytically.
- Explore the closed-loop quadrotor specifications and model the quadrotor experimentally.
- Develop an adaptive controller and realise AMPC (Adaptive dynamic matrix control - ADMC) based on the quadrotor system model obtained.
- Simulate AMPC (ADMC) pick-and-place and make evaluations against self-tuned adaptive control and also PID control.
- Deploy ADMC on the physical model and carry out experiments.
- Evaluate the quality of the deployed controller according to some experiments.

1.8 Literature Review

This research focuses on the development of a controller for pick-and-place aerial manipulation using a quadrotor. Literature of interest will contain information about aerial manipulation in general, control design, trajectory tracking and system modelling.

1.8.1 Control of aerial manipulators for pick-and-place

Kim et al. [8] adopt an adaptive sliding mode controller to solve the pick-and-place problem. This is realised by a quadrotor with a 2 degree of freedom (DOF) actuated

arm to pick and place an object. In this work, the quadrotor and the actuated arm are modelled in a combined system model and experiments reveal that the adaptive sliding mode controller performs satisfactorily.

Garimella et al. [9] develop a nonlinear coupled model for a quadrotor with a 2 DOF manipulation arm. Unconstrained MPC is proposed (prediction horizon 2 to 5 seconds) and employed for trajectory tracking on an Odroid XU+E computer. An onboard vision system is then used to locate the object to be picked up. Experiments reveal satisfactory tracking, validated by an optical motion tracking system. Possible improvement, however, is in the efficiency of the optimisation control algorithm for which a fast optimal control algorithm is proposed as a possible solution. Ghaffar et al. [10] investigate model reference adaptive control for a quadrotor designed to pick up an unknown load. When combined with a linear quadratic regulator (LQR) the adaptive controller is capable of ensuring a stable tracking performance of the loaded quadrotor. Experiments are carried out on a fixed test-bench.

The most popular load-lifting technique is 'grasping', sometimes referred to as gripping. Pounds et al. [11] address quadrotor grasping proposing proportional integral differential (PID) control to reject the disturbances introduced by the load. Experiments are carried out using an aerial manipulator testbed. Results reveal that higher load mass had to be aligned with the centre of mass of the quadrotor as closely as possible otherwise the PID would not meet the required stability performance indices satisfactorily.

Ghadiok et al. [12] develop a quadrotor capable of picking and placing an object in global positioning system (GPS) denied environments. Using an onboard camera and simultaneous localization and mapping (SLAM) algorithms, the quadrotor is able to track the planned trajectory precisely while an infrared assisted secondary camera locates the target object. Nested PID controllers are implemented to stabilise the quadrotor and to navigate the system. Heredia et al. [13] design a back-stepping based controller for an octocopter with an actuated manipulation arm. A coupled nonlinear dynamic model is proposed and indoor experiments including lifting of sensors are successfully carried out. Mellinger et al. [14] achieve cooperative load transportation using four synchronised quadrotors. This work includes experiments carried out in the Vicon motion capture system but however, does not include automated load picking. The concept of cooperative grasping and trajectory tracking with a load is also studied in [15] and [16].

1.8.2 Quadrotor trajectory tracking

The load dropping task of interest in this research relies on the trajectory tracking capabilities of the quadrotor. Trajectory tracking with quadrotors is on-going research. Suicmez [17] proposes two methods for trajectory tracking; back-stepping and linear quadratic tracking (LQT) control. Simulation results show that both controllers ensure satisfactory tracking, with LQT tracking complex trajectories more accurately and more efficiently than back-stepping control. Bouabdallah et al. [18] and Madani et al. [19] also study back-stepping tracking. Back-stepping control,

though robust, depreciates in performance quality as the system model becomes more inaccurate.

In [20] robust trajectory tracking is realised by a dual-loop control scheme. The controller considers model inaccuracy, external disturbances and sensor noise, and an optimisation solver computes control action. Simulation and experimentation reveal that the controller is efficient. Lee et al. [21] propose and simulate a feedback linearisation based controller for trajectory tracking, obtaining satisfactory results. Benallegue et al. [22] combine feedback linearisation with a sliding-mode based observer for command tracking. Zou [23] proposes a PD controller for trajectory tracking and to improve performance, an optimal tracking-differentiator is added.

Liu et al. [24] propose sliding-mode control for trajectory tracking. Sliding-mode control ensures robustness to system uncertainties and external disturbances. To minimise the chattering effect introduced by sliding-mode control, a fuzzy controller is added to the scheme and simulation results show efficient tracking. In [18] sliding-mode trajectory tracking for a quadrotor is investigated and also in [25]. where sliding-mode control is combined with adaptation. Sliding-mode design requires the knowledge of the bound on the uncertainties, which in some cases poses a challenge in accurately estimating the uncertainty bounds. Adaptive and MPC trajectory tracking are examined in sections 1.8.3 and 1.8.4 respectively.

1.8.3 Adaptive Control

With adaptive control, it is possible to adjust the trajectory tracking control of the quadrotor according to the picked up load. Adaptive control, in general, is a solution to the problem of uncertain or varying plant dynamics and/or disturbances. Adaptive control is based online system identification via estimation of process parameters. High adaptive gains technically mean fast estimation convergence but may also cause instability and so it is typical to apply low gains, as found by Ghaffar et al. [10]. Monte et al. [26] propose an adaptive back-stepping controller for quadrotor trajectory tracking based on a nonlinear system model. The controller is based on L1 adaptive control and in simulation; compensation for model uncertainties and frequency-range-specific disturbance is achieved to satisfactory levels.

Adaptive control involves automated system identification by means of estimating parameters of the system model in real-time. Two estimator approaches are possible; the 'black-box' approach or the 'regressor' approach [27]. Adaptive control can be realised in both continuous [28, 29] and discrete-time [30, 31]. A method to stabilise a quadrotor against unknown parameters in the system model and against external disturbances is discussed in [32], where control is based on the use of a cerebellar model articulation controller (CMAC). A linear model is employed in this work in state space form. Neuro-adaptive control procedures are implemented to train the approximation weights online. The CMAC controller is able to efficiently return the quadrotor to the desired position after induced disturbances are applied.

In [33] an adaptive method based on sliding-mode control is proposed for quadrotor control. The goal is to design an adaptive control law based on sliding-mode control

without the knowledge of the bound of uncertainties/perturbations. This control strategy is successful as demonstrated by simulations but with the possibility of improvement in compensation of the errors in model estimation.

A block back-stepping controller is proposed for the attitude stabilisation of a quadrotor in [34]. Adaptive algorithms provide an estimation of the state space model errors (nonlinear model), as well as the disturbances and a Lyapunov function ensures exponential estimation convergence. From simulations, it is evident that the designed controller effectively maintains stability when momentary disturbances are applied. An adaptive integral back-stepping controller is proposed by Fang et al. [35] with the aim to complement any inaccuracy in the model with adaptation, to minimise compromise on the back-stepping controller (which degrades with model inaccuracy). Simulation shows high quality control, regardless of model uncertainties.

Adaptive neural network control of quadrotor is pursued by Boudjedir et al. [36]. The research proposes an adaptive sliding-mode control based on two neural networks for quadrotor stabilisation. The learning algorithms are obtained using the direct Lyapunov stability method. While sliding-mode control independently provides a systematic approach for maintenance of stability when the system model is inaccurate, it does have some drawbacks, i.e. the chattering phenomena and the need to have knowledge of the system dynamics. This is complemented by the Neural Network control part of the system. Results of this work show stable control of the quadrotor regardless of dynamic model inaccuracies. Based on simulations, the controller also maintains stability when disturbances are applied.

1.8.4 MPC

Adaptive control is a control approach rather than control law, making it flexible to the actual control law implemented (whether PID, pole-placement, back-stepping, MPC etc). MPC provides optimal control via minimisation of a cost function of error and has the ability to observe constraints which typically represent physical limitations in the system. In combination with adaptation, MPC should ensure the desired control quality for this research. One-step-ahead prediction (e.g. Minimum Variance approach) methods can be implemented, which are based on plant output at a single future instant in the prediction horizon [37]. Control is then chosen so that the predicted output matches the reference-point. Such methods are however, very sensitive to dead-time. To achieve high robustness, it is helpful to look into long-range model predictive techniques such as generalised predictive control (GPC), dynamic matrix control (DMC), identification and command (IDCOM) etc [30]. These however, introduce a heavier computation burden.

Alexis et al. [38] propose nonlinear MPC for quadrotor trajectory tracking. The controller is based on the piecewise affine (PWA) method and constraints are realised in accordance with physical limitations of the quadrotor such as maximum rotor rotational velocity. Simulation indicates efficiency in trajectory tracking and disturbance rejection. A long range model predictive controller is designed for a miniature remote-controlled helicopter in [39] based on a Linear Time Varying (LTV) model

approximated from the vehicle's non-linear dynamics. A fast custom convex optimization solver for the MPC iteration is developed and experiments conducted to compare MPC against a PID controller. Results indicate that the PID does not meet the quality indices of the MPC based controller.

An MPC scheme is proposed for attitude control in [40] for a quadrotor UAV. The predictive controller is able to stabilise the quadrotor model when induced wind disturbances are applied, however, when operating conditions change significantly, the predictive controller becomes vulnerable. To this problem, the authors suggest inclusion of on-line system identification to update the model in real-time.

In [41] MPC is implemented for the altitude control of a quadrotor employing a reduced linear model to improve efficiency of the controller by reducing computational burden. The controller is implemented on an embedded linux-based Gumstix flight controller for experiments and results show satisfactory response, although the experiments are limited to a narrow range of operating conditions. Kim et al. [42] propose MPC for quadrotor control, with the aim to improve efficiency by employing Weighted Forgetting on constraints in order to prioritise and base computation only on data that has more effect on the system. Simulations show satisfactory response. One drawback with MPC is that if the internal prediction model deviates from the actual process, performance of the controller degrades. We expect lifting a load to change the dynamics of the quadrotor (system model) significantly and so, for efficient MPC, a method to keep the internal model updated may be necessary.

1.8.5 AMPC

AMPC or Learning-Based Model Predictive Control (LBMPC) is the realisation of MPC in a manner which involves updating the MPC internal model as system dynamics change. Bouffard et al. [43] apply LBMPC on a quadrotor for control and trajectory tracking. A nonlinear model is employed in state space. The controller ensures convergence and robustness efficiently regardless of changing process dynamics because the controller learns (based on statistical algorithms) and updates the MPC internal model according to the changing dynamics. This remains true even when the learning algorithms produce inaccurate models.

Aymes [44] also applies MPC to an adaptive scheme for the stabilisation of a marble on a rail modelled into a simplified linear MPC transfer function model. The predictive controller used is GPC based and the model is represented by an AutoRegressive-Moving Average with Exogenous Terms (ARMAX) structure. The controller is enhanced to utilise a supervisor to improve robustness to modelling errors and experiments reveal large stability margins and is implemented on a TMS320C50 digital signal processor. Chowdhary [45] investigate AMPC for a UAV subsystem. The nonlinear dynamics of the system are reduced into a linear model and simulation shows effective control. The linearised model is represented in discrete state space form. Rahideh et al. [46] also investigate AMPC based on a twin-rotor aerodynamic test bed. Again, the nonlinear dynamics are linearised and represented by discrete state space models. Yaw and pitch tracking experiments were carried out and results showed efficiency in the method.

1.8.6 System modelling

AMPC is based on an internal model for output prediction. It is therefore necessary to have a reliable model of the quadrotor. There are two approaches to modelling the dynamics of a quadrotor UAV system. The first is analytical, based on dynamic equations (Newton-Euler or Euler-Lagrange formalisations) and the second is experimental, based on input-output data. Balas [47] takes the first approach to system modelling. The quadrotor dynamics are modelled based on kinematic and dynamic equations using Newton-Euler formalisation. Bouabdallah has studied the OS4 miniature quadrotor [48], initially, developing simple models describing the vehicle in hover flight based on kinematic and dynamic equations. The rotor system is modelled experimentally and system parameters are derived from a computer aided design (CAD) model. Ahmed et al. [49] and Nicol et al. [50] have also taken the analytical approach to system modelling based on Newton-Euler formulation. The Lagrange approach, as detailed by Jithu et al. [51], models the quadrotor as a complete (whole) instead of a system of separate components. Xu et al. [52] also take the Lagrange approach to modelling a quadrotor. Both formulations produce similar end models, the Newton-Euler approach being relatively more flexible.

The experimental approach to quadrotor system modelling is done on/in quadrotor test-beds [53]. There are two kinds of quadrotor test-beds; one based on ready-to-fly quadrotor prototypes [53, 54] and the other based on customised platforms in/on which the quadrotor prototype is placed and tested [55]. The combination of both kinds of test-beds should give high accuracy (for higher cost) and safety as proposed in [56] where a commercial quadrotor, the Vicon motion capture system and a control station are integrated within an experimental field. The Vicon motion capture system comprises eight cameras and a server from which data is logged in real-time into MATLAB. From the motion tracking system, it is possible to receive position and orientation information of the quadrotor. A mathematical model of the quadrotor is then derived via system identification. The mathematical model obtained resembles the physical dynamic quadrotor system closely.

In [57], research focuses on deriving a mathematical model of a quadrotor based on a prototype quadrotor. Data is logged onto a secure digital card during flight and is imported post-flight, into MATLAB for Single-Input Single-Output (SISO) system identification based on Comprehensive Identification from Frequency Response (CIFER). The quadrotor system, in its closed-loop structure, is modelled experimentally in [58], Stanculeanu et al. and PID controllers are setup to stabilise the quadrotor to hover before the data acquisition experiment, in which control data is stored on an SD card. Khorani et al. [59] use a genetic algorithm to process data from the gyros and accelerometers of the quadrotor. Experimental quadrotor system modelling is also investigated in [60] and [61]. Particularly, Yang et al. [60] obtain second-order system models based on the ARMAX structure which is analysed to adequately characterise the closed-loop response of the system. While models obtained experimentally are generally more direct and practical, these models are typically reduced to lower order systems, posing a drawback of possible acute degradation in accuracy. The same applies to analytical models which are typically linearised.

1.8.7 Experimental Validation

The full AMPC controller must be implemented and validated via some kind of load dropping experiments. The idea is to observe if the quadrotor follows the reference command accurately. Vicon motion capture system based experiments offer a reliable method for validation [14, 56]. While the Vicon motion capture system provides a reliable test platform, it does limit the experiments to an indoor setting. However, outdoor conditions can be induced in the environment with additional Vicon systems [56].

Less restrictive validation is possible if a data acquisition system is employed to capture position information from onboard sensors. Befus [62] takes such an approach, where positional data is recorded from ultrasound range sensors and the inertial sensor. This method is successful although the experiments are still limited to a confined space (indoor). Muller et al. [63] explore a similar approach and achieve efficient tracking with an addition of airflow sensors. Jose et al. [64] also achieve low localisation errors by using a radio transmitter and determining distance based on signal time delay.

1.8.8 Feasibility

The feasibility of AMPC for quadrotor control is substantially proven. It is feasible to update the prediction model of the adaptive model predictive controller in real time, for improved quadrotor control [43]. It is worth mentioning that adpative MPC becomes feasible if the internal model is structured to accommodate the applied recursive algorithms. Typically, simple models result in improved recursion, although the model, in its simplicity, must maintain ability to describe the system dynamics. Persistent excitation for the model, is another aspect that affects feasibility of adaptive MPC, as discussed in Chapter 3 of this work. Both linear [45] and non-linear [43] models have yielded satisfactory results, and in all cases, seemingly acceptable compensation for model inaccuracies. On the scale of implementation and experimentation, the feasibility of AMPC for quadrotor control is heavily dependent on the computational capacity of the actual hardware employed.

1.8.9 Computational speed

Due to the high computational demand of AMPC, any implementation and experimental exercises require hardware equipment that is capable of meeting the computation need [46]. In particular, the MPC algorithm will solve the an optimisation problem at each sample time. On that note still, acquisition of relevant information from the quadrotor control system sensors (accelerometers, gyros, barometer and GPS) must be carried out, as well as execution of the AMPC control action through the motor control modules. If the MPC iteration exceeds the loop duration (based on the frequency specification of the processing computer), the controller is bound to fail [44]. It is necessary to ensure that the available hardware is able to provide the required computational speed to complete all iterations.

1.8.10 Stability

Control system stability can be defined as the ability of the system to maintain constant state unless influenced by some external action, which case the system returns to some constant state when the external influence is removed [3]. The quadrotor system is inherently unstable [18], stemming from the fact that the linearised system model can be represented by transfer functions of second order with two poles on the origin as explained in the next chapter. Stability has been successfully achieved through the implementation of various control strategies ranging from PID [12], to linear quadratic [17], back-stepping [19], sliding-mode [24] and so on. It is common that finite time optimal controllers, which include MPC, do not necessarily always result in stable closed-loop systems. MPC relies on some iterative solution of an optimal control problem. Typically, to ensure stability of MPC, the idea is to employ strategies on the basis of some terminal cost function and some terminal constraint region. It is typical to impose state constraints and also control constraints to ensure stability. MPC stability is known to be sensitive to the lengths of the control and prediction horizons too, where too short such horizons may lead to instability. Model uncertainty as well, is a common contributor to unstable MPC.

1.9 Summary

In this chapter, a background to unmanned aerial vehicles was given and to the research done in this work. It is clear from this chapter that this research investigates the application of Adaptive dynamic matrix control (ADMC) to quadrotor control given varying load. Varying load eercises translate to pick-and-place operations with quadrotors, which are important for applications such as delivery of medical supplies or even general courier. To perform this efficiently and safely, several quality requirements must be met with regards for example to control, air frame design, electronics etc. This research investigates the control aspect. It is clear from the literature review, that several approaches can be taken for quadrotor system modelling and control design. Each approach has disadvantages and advantages depending on the point of view and the nature of the problem to be solved. From the literature review, AMPC stands out for the application investigated in this research, one which demands adaptation to changing plant dynamics and optimal reference tracking.

Since the majority of the work done on ADMC does not take computational expense and implementation (especially for such systems as multi-rotors) into account, this work looks into ADMC from theory to implementation. Given that ADMC typically involves frequent switching from one model to another, with the intention to improve controller efficiency, this work proposes the use of a single adjustable step response model for different disturbance influence, hence limiting switching. This follows several simulations to analyse how model parameters are affected by disturbance. This is supported by a Supervisor which monitors system behaviour and then decides how to adjust the disturbance step response model. The expected profits of using a supervisor are code simplicity and hence efficiency. System modelling

is discussed in Chapter 3, ADMC in Chapter 4 and implementation of ADMC in Chapter 5.

Chapter 2

SYSTEM MODELLING AND IDENTIFICATION

Adaptive model predictive control is based on an internal model, which is updated in real-time according to learning algorithms (adaptive laws) and is used to predict system output. The start point of this research is therefore the acquisition of a system model. The system model is developed experimentally through system identification of the fully integrated and ready-to-fly quadrotor system. The outcome is a custom model based on practical inputs and outputs and representing the dynamics of the physical quadrotor under investigation.

2.1 Analytical Modelling

Figure 2.1 shows the structure of the Quadrotor and demonstrates the inertial (right) and body (left) axes which are useful in characterising flight principles of the quadrotor.

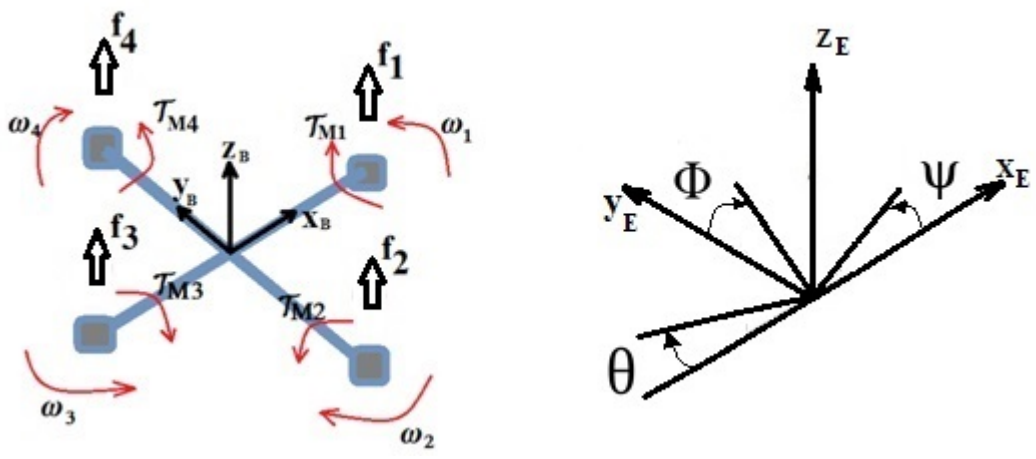

FIGURE 2.1: Structure of the Quadrotor and the inertial and body axes

The angular rates of the four rotors determine the state of the UAV in a manner described next. The two pairs of rotors (1 and 3) and (2 and 4) spin in opposite

directions, such that the torque effect resulting from the rotors cancels and stable operation is made possible. The ith rotor spins with angular velocity ω_i and produces a lift force f_i and a torque τ_i in the opposite direction as ω_i.

Analytical modelling of a quadrotor, as explained in the next, is characterised by kinematic and dynamic equation derivation [47, 48, 49, 50], following the Newton-Euler formalisation, as stated in Chapter 2.

2.1.1 Kinematics of Quadrotor

The inertial frame, whose origin is the centre of mass of earth, is marked by x_E-y_E-z_E and the body frame, whose origin is the centre of mass of the quadrotor, is marked x_B-y_B-z_B. Linear position (expressed in the inertial frame) is defined in reference to vertical (z), longitudinal (x) and lateral (y) displacement by equation 2.1

$$\mathbf{X} = \begin{bmatrix} x, y, z \end{bmatrix}^T \tag{2.1}$$

Attitude (expressed in the inertial frame) is defined by three Euler Angles; roll (ϕ) rotation angle about x, pitch (θ) about y and yaw (ψ) about z, by equation 2.2

$$\mathbf{\Theta} = \begin{bmatrix} \phi, \theta, \psi \end{bmatrix}^T \tag{2.2}$$

f_i represents the thrust produced by the ith rotor, τ_{Mi} represents the torque resulting from the ith rotor and ω_i represents the rotational velocity of the ith rotor.

Linear and angular velocities (expressed in the body frame) are represented by u, v, w and p, q, r (in the order of roll - pitch - yaw) respectively (equations 2.3 and 2.4).

$$\mathbf{V} = \begin{bmatrix} u, v, w \end{bmatrix}^T \tag{2.3}$$

$$\mathbf{\Omega} = \begin{bmatrix} p, q, r \end{bmatrix}^T \tag{2.4}$$

The existence of two different coordinates frames means that a transformation is necessary in each case, to convert one frame to the other for any meaningful relationships to be deduced. This transformation is done by using Direction Cosine Matrices. In this way, relationship between the velocity vectors can be presented as equation (2.5). The reason behind the use of direction cosine matrices is that direction cosine matrices allow the representation of some set of orthonormal-type vectors in terms of another different set of vectors.

$$\begin{cases} \dot{\mathbf{X}} = \mathbf{R}(\mathbf{\Theta})\mathbf{V} \\ \dot{\mathbf{\Theta}} = M^{-1}(\mathbf{\Theta})\mathbf{\Omega} \end{cases} \tag{2.5}$$

$\boldsymbol{R}(\boldsymbol{\Theta})$ and $\boldsymbol{M}^{-1}(\boldsymbol{\Theta})$ are the transformation matrices for the translation (equation 2.6) and rotation (equation 2.7) velocities between the body and the earth axes respectively. The angular velocity ($\boldsymbol{\Omega}$) is therefore a vector pointing along the axis of rotation, while $\dot{\boldsymbol{\Theta}}$ is only the time derivative of yaw, pitch and roll.

$$\boldsymbol{R}(\boldsymbol{\Theta}) = \begin{bmatrix} c\psi c\theta & c\psi s\theta s\phi - s\psi c\phi & c\psi s\theta c\phi + s\psi s\phi \\ s\psi c\theta & s\psi s\theta s\phi + c\psi c\phi & s\psi s\theta c\phi - c\psi s\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \tag{2.6}$$

where where $c\cdot = cos(\cdot)$ and $s\cdot = sin(\cdot)$ and $\mathbf{R}$ is a rotation matrix from the body frame to the inertial frame, one which is orthogonal such that $R^{-1} = R^T$ which conveniently represents the rotation matrix from the inertial frame to the body frame.

$$\boldsymbol{M}(\boldsymbol{\Theta}) = \begin{bmatrix} 1 & 0 & -s\theta \\ 0 & c\phi & c\theta s\phi \\ 0 & -s\phi & c\theta c\phi \end{bmatrix} \tag{2.7}$$

The angular velocity transformation matrix from the inertial frame to the body frame is $\mathbf{M}$ and from the body frame to inertial is $\mathbf{M}^{-1}$, where:

$$\boldsymbol{M}^{-1}(\boldsymbol{\Theta}) = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & \frac{s\phi}{c\theta} & \frac{c\phi}{c\theta} \end{bmatrix} \tag{2.8}$$

2.1.2 Dynamics of a Quadrotor

The quadrotor is modelled for near-hover operation. First, the quadrotor is imagined to be a free body with gravitational force (opposite to z_B) and thrust force acting on the quadrotor, opposite to gravitational force. The rotor-speeds ω_1, ω_2, ω_3 and ω_4 control the quadrotor according to the four consequent physical parameters; total thrust force (u_1) in the vertical direction and rolling torque (u_2), pitching torque (u_3) and yawing torque (u_4) in the corresponding orientation direction. The relationship between the thrust and angular rotational velocity of a rotor is known to be of the form (2.9)

$$f = k\omega^2 \tag{2.9}$$

where k is the lift coefficient of the rotor. This comes from a series of mathematical equations, beginning with equation (2.10), which expresses the torque produced by a brushless motor as

$$\tau = K_t(I - I_0) \tag{2.10}$$

where τ is the motor torque, K_t is the torque scaling constant, I is the input current and I_0 is the current drawn when the load is zero. The voltage across the motor is

the sum of the back-EMF (Electro-Motive Force) and losses due to resistance (2.11)

$$V = IR_m + K_v\omega \tag{2.11}$$

where V is the voltage drop across the motor, R_m, the resitance of the motor, K_v is a proportionality constant for the back-EMF caused per RPM and ω is the angular velocity of the motor. It is then possible to use this description to determine the power consumed by the motor (2.12)

$$P = IV = \frac{(\tau + K_t I_0)(K_t I_0 R_m + \tau R_m + K_t K_v \omega)}{K_t^2} \tag{2.12}$$

If the resistance of the motor is assumed to be negligible, then the power becomes proportional to the angular velocity (2.13)

$$P \simeq \frac{(\tau + K_t I_0) K_v \omega}{K_t} \tag{2.13}$$

Since I_0 is the current when there is no load, and thus rather small, it is assumed that $K_t I_0 \ll \tau$. This further reduces the power equation to (2.14)

$$P \simeq \frac{K_v}{K_t}\tau\omega \tag{2.14}$$

Now, the power thus will generate lift for the quadrotor. Based on the principle of conservation of energy, the energy the motor spends in some time period will equal the force generated on the propeller multiplied by the distance that the air displaced travels ($P \cdot dt = f \cdot dx$). It follows that

$$P = f\frac{dx}{dt} \tag{2.15}$$

which can be written as

$$P = f v_h \tag{2.16}$$

v_h is the air velocity at hover, and it is assumed that the air in the surrounding environment is stationary relative to the quadrotor such that, based on the momentum theory [48], the equation for hover velocity is

$$v_h = \sqrt{\frac{f}{2\rho A}} \tag{2.17}$$

where ρ is the density of the surrounding air and A is the area swept out by the rotor. The equation for power can therefore be written as the equation below. Torque (τ) is proportional to rotor thrust f by some constant. If this constant is denoted by K_τ, the following equation is derived:

$$P = \frac{K_v}{K_t}\tau\omega = \frac{K_v K_\tau}{K_t} f\omega = \frac{f^{\frac{3}{2}}}{\sqrt{2\rho A}} \tag{2.18}$$

It now comes out that f is proportional to the square of angular velocity of the motor as in equation (2.19), which brings about equation (2.9).

$$f = \left(\frac{K_v K_\tau \sqrt{2\rho A}}{K_t} \omega \right)^2 = k\omega^2 \tag{2.19}$$

Consequently, total thrust (**T**) is (2.20)

$$\mathbf{T} = k \sum_{i=1}^{4} \omega_i^2 \tag{2.20}$$

The relationship between the torque and angular rotational velocity of a rotor can also be determined. Torque is required to keep the propeller spinning. This torque causes angular acceleration and overcomes the frictional drag force. Fluid dynamics gives frictional (in this drag) force as (2.21)

$$F_{drag} = 0.5\rho C_D A v^2 \tag{2.21}$$

where ρ is the density of the surrounding fluid, A is the reference area (propeller cross-section in this case), C_D is a constant and v is flow velocity. It therefore follows, assuming all the force is applied at the tip of the propeller, that torque of the rotor due to drag is given by

$$\tau = 0.5R\rho C_D A v^2 = 0.5R\rho C_D A(\omega R)^2 \tag{2.22}$$

The relationship between the torque and angular rotational velocity of a rotor can then be written as (2.23)

$$\tau = b\omega^2 \tag{2.23}$$

where b is the drag coefficient of the rotor.

Torque in the z axis for the ith motor can be written in detail as

$$\tau_z = b\omega^2 + I_M \dot{\omega} \tag{2.24}$$

where I_M is the moment of inertia about the z axis and $\dot{\omega}$ is the angular acceleration of the propeller, which at hover can be approximated to zero. A new expression for the torque is therefore (2.25)

$$\tau_z = (-1)^{i+1} b\omega_i^2 \tag{2.25}$$

where $(-1)^{i+1}$ is positive for the ith propeller if the propeller is spinning clockwise and negative if spinning anticlockwise. The total torque about the z axis is given by the sum of all the torques from each propeller (2.26)

$$\tau_\psi = b\omega_1^2 - b\omega_2^2 + b\omega_3^2 - b\omega_4^2 \tag{2.26}$$

Arbitrarily selecting motors number 1 and 3 to be on the roll axis, the roll torque can be expressed as (2.27)

$$\tau_\phi = \sum r \times f = L(k\omega_1^2 - k\omega_3^2) = Lk(\omega_1^2 - \omega_3^2) \tag{2.27}$$

where L is the distance from the center of the quadrotor to any rotor and for simplicity. Likewise, pitch torque can be expressed as (2.28)

$$\tau_\theta = Lk(\omega_2^2 - \omega_4^2) \tag{2.28}$$

With equations (2.20, 2.26, 2.27 and 2.28) it is possible to consider inputs to the quadrotor according to equation (2.29)

$$\begin{cases} T = u_1 = k(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \\ \tau_\phi = u_2 = Lk(\omega_1^2 - \omega_3^2) \\ \tau_\theta = u_3 = Lk(\omega_2^2 - \omega_4^2) \\ \tau_\psi = u_4 = b(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{cases} \tag{2.29}$$

Newton's Law can then be applied, on the basis that, the force ($m\dot{\mathbf{V}}$), required to accelerate a quadrotor of mass (m) at an acceleration of ($\dot{\mathbf{V}}$) and centrifugal force [$\boldsymbol{\nu} \times (m\mathbf{V})$] is equal to the sum of the gravitational force ($\mathbf{R}^T\mathbf{g}$) and total thrust ($\mathbf{T}$). The thrust vector $\mathbf{T}$ in the z_B-axis direction and is the sum of all rotor lift forces (2.30)

$$\boldsymbol{T} = \begin{bmatrix} 0 \\ 0 \\ u_1 \end{bmatrix} \tag{2.30}$$

A torque vector for torques u_2, u_3 and u_4 in the direction of corresponding body frame angles can be written as (2.31)

$$\boldsymbol{\tau} = \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{2.31}$$

In the body frame, the force required for the acceleration $m\dot{\mathbf{V}}$ of mass m with some centrifugal force ($\boldsymbol{\nu} \times (m\mathbf{V})$) is equal to the gravity $\mathbf{R}^T\mathbf{g}$ and the total thrust of the rotors. The equation (2.32) can then be derived:

$$m\dot{\mathbf{V}} + \boldsymbol{\nu} \times (m\mathbf{V}) = \mathbf{R}^T\mathbf{g} + \boldsymbol{T} \tag{2.32}$$

In the inertial frame, only the gravitational force and the magnitude and direction of the thrust contribute to the acceleration of the quadrotor. Centrifugal force can be nullified given the size of earth and equation (2.33) arises

$$m\ddot{\mathbf{X}} = \mathbf{g} - \mathbf{R}\boldsymbol{T} \tag{2.33}$$

where $-\boldsymbol{RT}$ means direction of z is downward. This then leads to equation (2.34)

$$\begin{cases} \ddot{x} = -(c\psi s\theta c\phi + s\psi s\phi)\dfrac{u_1}{m} \\ \ddot{y} = -(s\psi s\theta c\phi - c\psi s\phi)\dfrac{u_1}{m} \\ \ddot{z} = -(c\theta c\phi)\dfrac{u_1}{m} + g \end{cases} \tag{2.34}$$

Euler angular rates are related to body angular rates with a rotational matrix as in equation (2.35)

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & \dfrac{s\phi}{c\theta} & \dfrac{c\phi}{c\theta} \end{bmatrix} \cdot \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.35}$$

where $t\cdot = tan(\cdot)$. Differentiating (2.35) with respect to the Euler parameters gives (2.36)

$$\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & \dot{\phi}c\phi t\theta + \dot{\theta}\dfrac{s\phi}{c^2\theta} & \dot{\phi}s\phi c\theta + \dot{\theta}\dfrac{c\phi}{c^2\theta} \\ 0 & -\dot{\phi}s\phi & -\dot{\phi}c\phi \\ 0 & \dot{\phi}\dfrac{c\phi}{c\theta} + \dot{\phi}s\phi\dfrac{t\theta}{c\theta} & -\dot{\phi}\dfrac{s\phi}{c\theta} + \dot{\theta}c\phi\dfrac{t\theta}{c\theta} \end{bmatrix} \cdot \begin{bmatrix} p \\ q \\ r \end{bmatrix} + \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & \dfrac{s\phi}{c\theta} & \dfrac{c\phi}{c\theta} \end{bmatrix} \cdot \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} \tag{2.36}$$

Assuming the structure of the quadrotor is symmetrical, the inertial matrix is as (2.37)

$$\boldsymbol{I} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{2.37}$$

Now, torque is a product of moment of inertia and angular acceleration ($\boldsymbol{\alpha}$) there equation (2.31) can be extended to equation (2.38)

$$\boldsymbol{\tau} = \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} = \boldsymbol{I\alpha} \tag{2.38}$$

On the other hand, angular acceleration can also be expressed as the time derivative of angular velocity such that an equation (2.39) is obtained

$$\frac{d(\boldsymbol{I\Omega})}{dt} = \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} = \boldsymbol{I} \cdot \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} + (\boldsymbol{I\Omega} \times \boldsymbol{\Omega}) \tag{2.39}$$

The term $(\boldsymbol{I\Omega} \times \boldsymbol{\Omega})$ can be neglected in this case assuming r is small given that $(I = r^2 m)$. This leads to equation (2.40)

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \boldsymbol{I^{-1}} \cdot \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{2.40}$$

The expression in equation (2.40) can then be inserted into equation (2.36).

The system can be linearised, by Taylor's Method, at hover condition where it is assumed $(\dot{\phi} = \dot{\theta} = \dot{\psi} = 0)$ and $(u_2 = u_3 = u_4 = 0)$ and the expression for lift force is presented by equation (2.41)

$$u_1 = \frac{m(g - \ddot{z}_{hov})}{c\theta_{hov} c\phi_{hov}} \tag{2.41}$$

The linear model obtained is of the form (2.42) and (2.43)

$$\begin{cases} \ddot{x} = -g\theta \\ \ddot{y} = g\phi \\ \ddot{z} = -\dfrac{u_1}{m} \end{cases} \tag{2.42}$$

$$\begin{cases} \ddot{\phi} = \dfrac{u_2}{I_{xx}} \\ \ddot{\theta} = \dfrac{u_3}{I_{yy}} \\ \ddot{\psi} = -\dfrac{u_4}{I_{zz}} \end{cases} \tag{2.43}$$

It follows that the transfer function of $(z, \phi, \theta and \psi)$ is of second order with two poles on the origin, and can be estimated as:

$$G(s) = \frac{k}{(t_1 s + 1)(t_2 s + 1)} \tag{2.44}$$

Several other aerodynamic effects (effect of angle of attack, blade flapping, drag force etc) [48, 50] could be included in the model but this is not in the interest of this research as the system is modelled experimentally, in the next part of this chapter. The experimental model is the model that is adopted for control design.

2.2 System Identification for the Quadrotor

The Newton-Euler method is able to provide a system model for the quadrotor. The model however, is a set of nonlinear equations and which describe the quadrotor dynamics by input signals of force and moment, instead of trajectory input commands, as appropriate to this study. The model is typically linearised about the hover-stabilisation point to produce linear models which are in some cases sufficient

in modelling the system and also control design [45, 47, 57, 60]. In this research, the subsystems of interest are modelled experimentally through system identification. The result, in each case, is a linear model with input signals as position commands. To consider the real-system non-linearities, relevant tools are therefore used such as signal saturation and time delays.

The flight controller used on the experimental quadrotor has the capability to log flight data. This function is used to record actual flight data in real-time, during a flight, for example, sampled readings of altitude (height above sea level) of the physical quadrotor in meters, recorded per unit time. In the case of altitude, this data is measured by a real barometric sensor on board the quadrotor and this is the data that is used for system identification, in this case, for the vertical control subsystem. A log file is created automatically for each flight and can be analysed by the Mission Planner software or otherwise converted into a spreadsheet and then imported into suitable software such as Matlab for system identification; sample input-output data is presented in Appendix A.

2.2.1 Purpose of system identification

This subsection spells out the motivation behind system identification, for this work. The reason why flight data is collected for system identification is the need for an experiment-based model structure, with some parameters which can be adjusted to represent different operating conditions of the system. This model structure, will be adopted for adaptive control design (chapter 4). It is not in the interest of this work to identify particular model parameters such as thrust factor or moment of inertia and so on.

2.2.2 Analytical approach

Two approaches are possible for the system identification, i.e. system identification based on real flight experimental data or based on Simulink model simulation data (from running the available Newton-Euler derived analytical model). As will be explained later, the main goal is to obtain an experiment-based model for quadrotor, which is identified from real flight input-output data. Obtaining this data safely requires that the quadrotor be stable for flight control hence modelling is typically for the closed-loop case [58, 60]. In this way, the quadrotor is PID stabilised to achieve more practical modelling.

Linearity and non-linearity

With respect to linearity and non-linearity, the conception is that when the quadrotor is operated near the hovering condition, a linear model will suffice. Then, for hard acrobatic-type operation, outside the linear region, the linear model typically begins to fall short [56]. The hover operation is of interest in this work. To account for non-linearities of the actual quadrotor system, signal saturation utilities and time delays

are used. The models identified for this work, are characterised by the following inputs and outputs:

1. Input control signal to longitudinal translation (x).
2. Input control signal to lateral translation (y).
3. Input control signal to vertical translation (z).

Input control signal in this case refers to step inputs for example 'fly 2 meters vertically.' Figure 2.2, shows the layout of the system, illustrating that the quadrotor is controlled only by managing the rotational speeds of the four rotors.

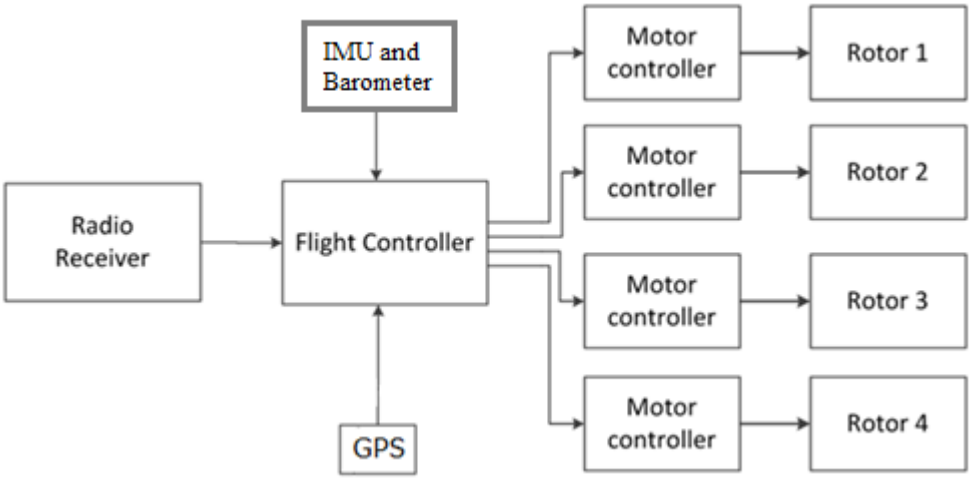

FIGURE 2.2: Layout of the experimental system

Resolving motor inputs from set-point input

In order to understand how the four motor inputs are resolved from the input control commands, it is necessary to consider the equations in section 2.1.2. These expressions translate to the illustration given by the 'Input block' under the 'Simulink model' subsection in section 2.2.2. Using the relevant simulation blocks, it is possible to realise the mathematical operations involved in modelling the quadrotor (Figure 2.6 and Figure 2.7).

Open-loop system

The open-loop uncontrolled system is unstable as seen from the linearised model, where the transfer functions of the model are of second order, with z, ϕ, θ, ψ directly affected by inputs u_1, u_2, u_3 and u_4, with two poles on the origin, making the system is inherently unstable. The transfer function takes the form:

$$G(s) = \frac{k}{s^2} \tag{2.45}$$

The system is closed and a PID controller designed for stabilisation.

Closed-loop system

The closed-loop system is illustrated by Figure 2.3 below. This illustration shows the coupled system structure of the quadrotor, and the character E_x represents the error signal for channel x i.e. the difference between the desired and measured values.

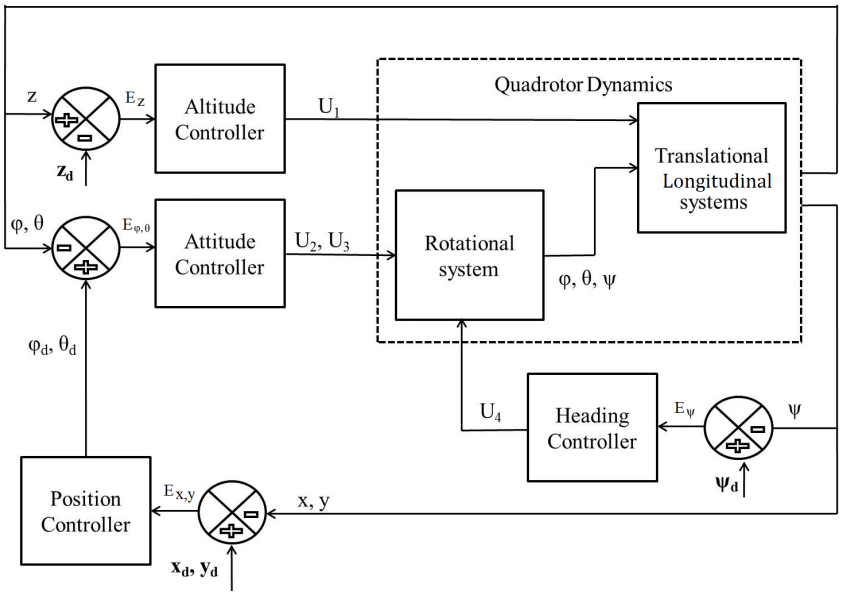

FIGURE 2.3: Coupled closed-loop system (adopted from (48))

On the diagram, the subscript of d on any variable depicts the desired value (reference input) of the variable. The coupled system model may be implemented in state-form as in [48] where the model is derived completely. Generally, a state vector describes not only the position of the quadrotor vehicle in space, but also the angular and linear velocities. At this point the plant is represented by the Newton-Euler based model derived at the beginning of this chapter.

The PID controller developed for the stabilisation of the quadrotor model is represented by Figure 2.4.

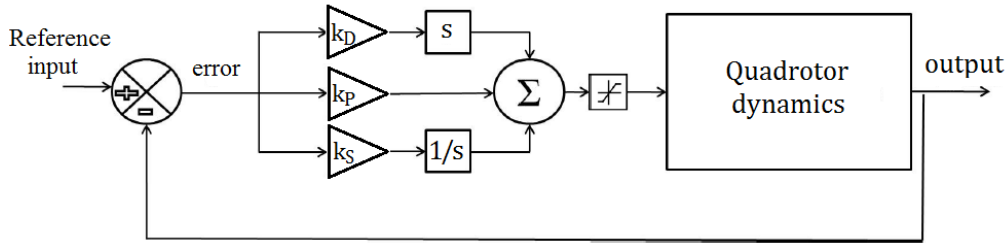

FIGURE 2.4: PID block diagram

The saturation block is useful in limiting the control signal to values of either 3 VDC or 5 VDC for interpretation by the motor controllers. The PID control law, of the altitude control system in particular, is represented by:

$$U_1 = k_P(Z - Z_D) + k_D(\dot{Z} - \dot{Z}_D) + k_I \int (Z - Z_D)dt \tag{2.46}$$

where k_P is the proportional gain, Z_D is the desired altitude, k_D is the derivative gain, $\dot{Z}_D$ is the desired rate of change of altitude and k_I is the integral gain.

The parameters for this case are given in Table 2.1.

TABLE 2.1: PID altitude control parameters

Parameter	Symbol	Value
Proportional	k_P	8.15
Integral	k_I	3.4
Derivative	k_D	0.28

Identification model derivation

System identification is necessary for this work because this results in a relatively more reliable model, representing accurate system dynamics. The discrete-time ARX model output is used, which is of the form (2.47). This model incorporates 'regressors', which as detailed in the next chapter, become the basis for parameter estimation. Given the parameter estimation method applied in this work, the ARX model is a sound solution, with a provision highly compatible with the control processes of the overall controller.

$$A(z)y(t) = B(z)u(t) + e(t) \tag{2.47}$$

The character u represents control action while e represents noise, then A(z) and B(z) are backward shift operator (z^{-1}) polynomials (considering zero input delay for simplification):

$$A(z) = 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_{n_a} z^{n_a} \tag{2.48}$$

$$B(z) = b_1 z^{-1} + b_2 z^{-2} + \cdots + b_{n_b} z^{n_b} \tag{2.49}$$

with the backward shift operator making $z^{-n}u(k) = u(k-n)$. This can be understood as:

$$y(t)+a_1y(t-1)+a_2y(t-2)+\cdots+a_{na}y(t-na) = b_1u(t)+b_2u(t-1)+\cdots+b_{n_b}u(t-nb) \tag{2.50}$$

where the functions $y(t-1), \cdots, y(t-na), u(t), \cdots, u(t-nb-1)$ are delayed output and input variables (regressors). If these functions are nonlinear then the ARX model is nonlinear. Having complex nonlinear functions is no guarantee that the

resultant model is more accurate. Several simulations are run until a satisfactory model is reached.

This model is described by Figure 2.5.

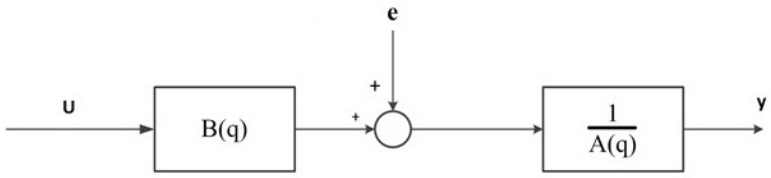

FIGURE 2.5: ARX model structure

A unique solution exists for the estimation of an ARX model (as with AR models). This therefore means that the solution is guaranteed to converge to the global optimum as the calculations are based on the least squares problem, which is a convex problem [5]. The AR model however, being time-series in nature, is only useful when modelling systems whose output is only dependent on previous outputs i.e. without including input or disturbances. The ARX, which considers previous output, input and also disturbance (stochastic) has the disadvantage on the other hand, that if the disturbance is not white noise, this might introduce bias to the estimation of the parameters of the model. It is important therefore, that the signal-to-noise ratio be kept high.

Simulation-data system identification

The procedure followed in system identification is similar for all subsystems. For this reason, only the vertical control subsystem is detailed. The analytical model is simulated and input-output data collected. The input-output data is then processed for system identification. The resultant model can then be compared and analysed against the model identified from flight experiment data. One possible way to obtain input-output data, is to write a simulator from the available equations of motion describing the dynamics of the system. The simulator uses Euler's method for solving differential equations to evolve the system state and functions to compute the rotation matrices. Alternatively, one can, from the analytical model, develop a Simulink model to represent the quadrotor system [18]. This method is preferable because it comes with the advantage to use relevant Simulink blocks, for example to analyse different model signals with ease.

For this research, the Simulink approach is taken. The ultimate goal is to model the experimental quadrotor, and to do this safely requires that the quadrotor be stabilised for flight control. Modelling in closed-loop is therefore common practice [58, 60] for quadrotors. For this reason, the quadrotor is PID stabilised to achieve more practical modelling especially from a safety perspective. To tune the PID controller, several methods including the Cohen-Coon method, Ziegler-Nichols rule [66] or Loop Shaping methods [67] can be used, based on objectives such as minimisation of overshoot or settling time. Better tuning can be achieved by using performance index optimisation techniques, which include optimisation of such indices as the

Integral of Squared Error (ISE), Integral of Absolute Error (IAE), Integral of Time Multiply Squared Error (ITSE) etc [68, 69]. As is the case with most optimization problems, this optimization is non-convex and so these optimisation techniques require some initial controller parameters, determined by one of the standard tuning methods such as the Cohen-Coon method. In this work, the PID tuner used to tune the PIDs is based on Ziegler-Nichols rule.

Simulink model

The analytical model is represented by a Laplace simulink model for the purposes of developing a model for running simulations to generate input-output data. As stated before, this input-output data is then processed to identify discrete-time ARX models for the next chapter. Figure 2.6 shows the Simulink model (adopted from [47] and [48]) for the input-output data modelling of the analytical model. The altitude control system is analysed and explained in detail. The controller block implements PID controllers which are tuned to stabilise the system. As stated earlier, DMC is based on an internal step response model, which must be stable.

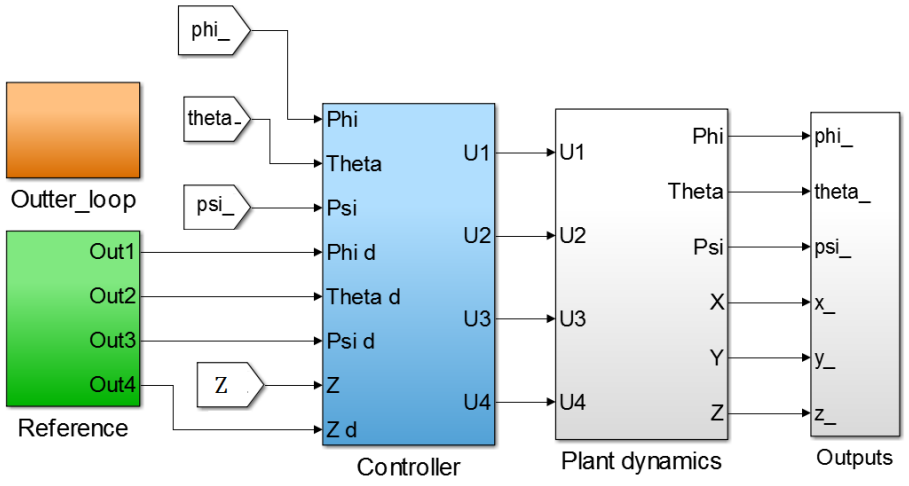

FIGURE 2.6: Simulink model

As stated in 2.2.2, the dynamics of the quadrotor are implemented in Simulink by translating the necessary mathematical expressions into the suitable blocks. Several representation options are possible, and with a complete model, signal limits can be set using the appropriate tools, to set for example, practical limits on angular rotational velocity of motors. The 'Plant dynamics' block contains a representation of the quadrotor plant, and is composed of two blocks, 'Block 1' and 'Block 2'. Appendix B gives the parameters used. Block 1 represents the derivation of angular accelerations and Block 2, translational accelerations, as illustrated by Figure 2.7.

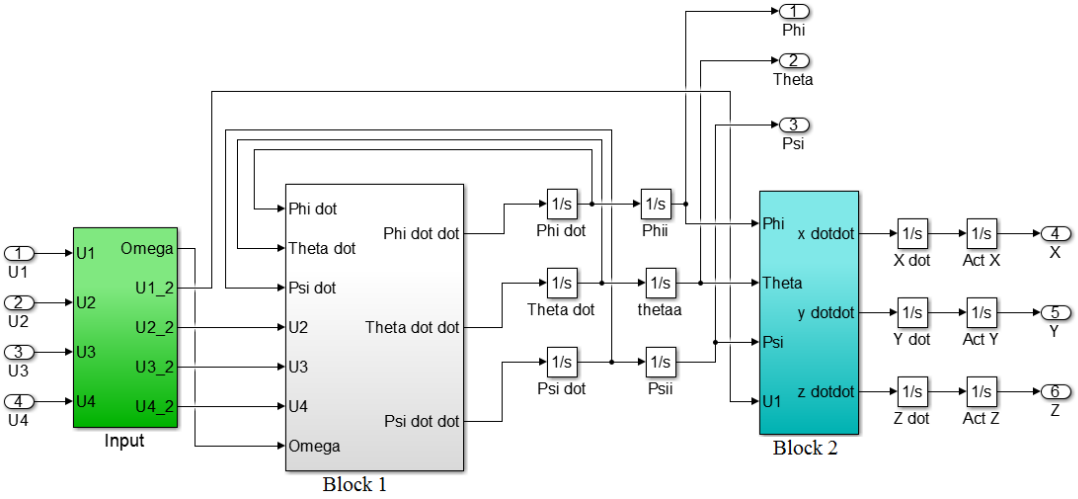

FIGURE 2.7: Plant dynamics

To complete the model, Figure 2.8 details Block 1 of the plant, while Figure 2.9 details Block 2.

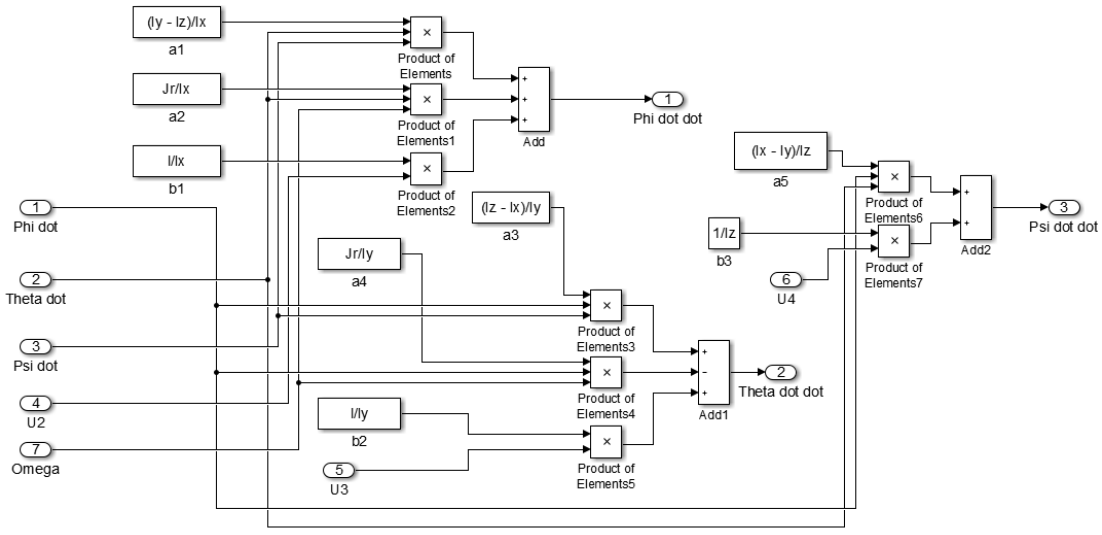

FIGURE 2.8: Angular accelerations block

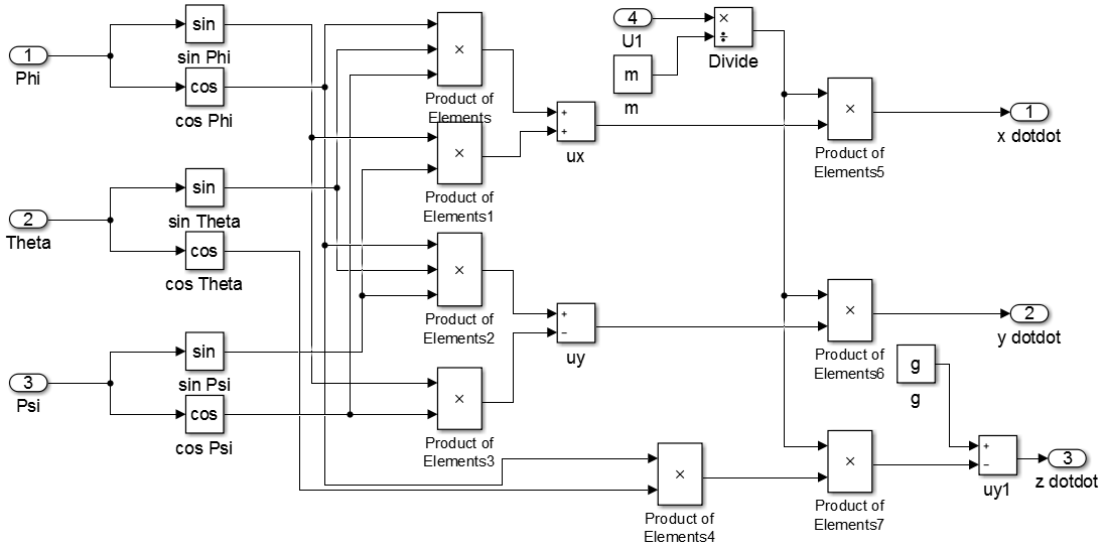

FIGURE 2.9: Translational accelerations block

Finally, the 'Input block' of the plant dynamics is represented by the figure below below, which finalises the quadrotor plant.

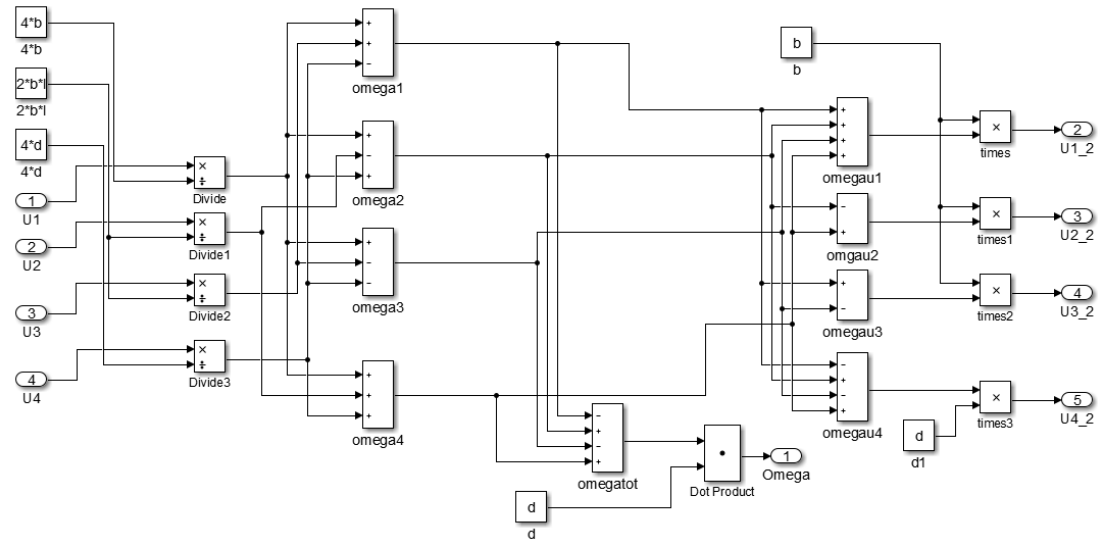

FIGURE 2.10: Input block

Going back to Figure 2.6, the reference signals can be realised by time functions, to allow commanding of complex trajectories. For the modelling exercise, step input and response is used and ramp input and ramp response for validation of the identified model. The block labelled '$Outter - loop$' receives reference signals in desired 'x' and 'y' positions and translates these signals into reference theta and phi signals which are used in the 'Reference' block. Typically this can be achieved by selecting

desired roll (ϕ) and desired pitch (θ) from (2.42) as follows:

$$\theta_d = \frac{-\ddot{x}_d}{g} \tag{2.51}$$

and

$$\phi_d = \frac{\ddot{y}_d}{g} \tag{2.52}$$

For this research, this translation is made possible by using a simple Gain Multiplier. This gain is derived based on maximum signal magnitudes. In this case, maximum displacement considered is 5 meters and maximum roll and pitch angle is 30 degrees (0.52 radians). The gain from x to θ or y to ϕ is therefore calculated as:

$$gain = \frac{0.52}{5} = 0.1(rad/meter) \tag{2.53}$$

The altitude system has a straight forward implementation from the analytical model. The parameters of the model are given in Table 2.2.

TABLE 2.2: Model parameters

Parameter	Unit	Symbol	Value
Mass	kg	m	0.55
Rotor wheelbase	m	l	0.23
Drag factor	–	d	$7.5 * 10^{-3}$
Thrust factor	–	b	$3.13 * 10^{-3}$

The step response is shown in Figure 2.11 and the ramp response in Figure 2.12.

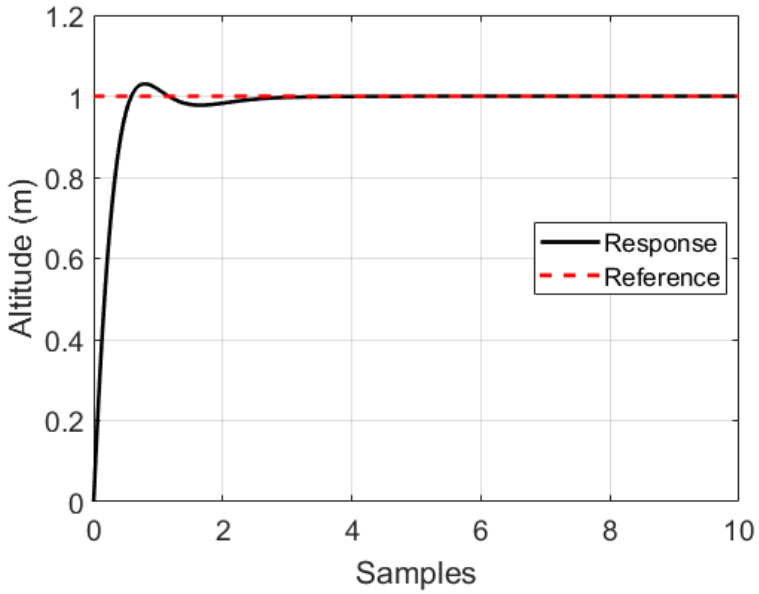

FIGURE 2.11: PID step response

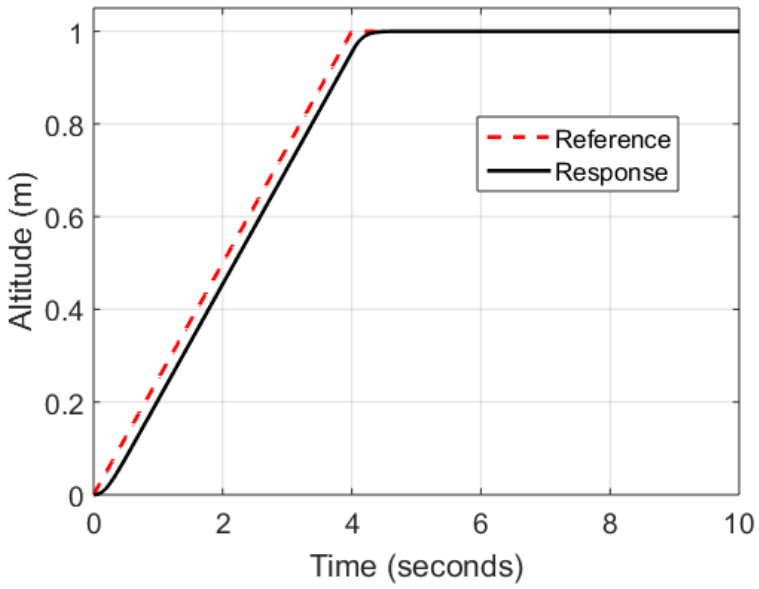

FIGURE 2.12: PID ramp response

Appendix B gives the model parameters of the physical quadrotor used in simulating the Simulink model. Step input and output data (0.01 seconds sampling interval) are used for system identification of the analytically obtained Simulink model. A sample of the data set from the Matlab workspace is shown in Figure 2.13.

	X	Y	Z	Xref	Yref	Zref	Clock
1	0	0	0	0	0	0	0
2	3.9477e-10	3.9477e-10	3.1606e-08	1.7500e-04	1.7500e-04	2.3750e-04	0.0100
3	9.0111e-09	9.0111e-09	1.7387e-06	3.5000e-04	3.5000e-04	4.7500e-04	0.0200
4	4.9485e-08	4.9485e-08	7.1961e-06	5.2500e-04	5.2500e-04	7.1250e-04	0.0300
5	1.5705e-07	1.5705e-07	1.8073e-05	7.0000e-04	7.0000e-04	9.5000e-04	0.0400
6	3.7341e-07	3.7341e-07	3.5666e-05	8.7500e-04	8.7500e-04	0.0012	0.0500
7	7.4357e-07	7.4357e-07	6.0995e-05	0.0011	0.0011	0.0014	0.0600
8	1.3139e-06	1.3139e-06	9.4776e-05	0.0012	0.0012	0.0017	0.0700
9	2.1310e-06	2.1310e-06	1.3753e-04	0.0014	0.0014	0.0019	0.0800
10	3.2410e-06	3.2410e-06	1.8968e-04	0.0016	0.0016	0.0021	0.0900
11	4.6891e-06	4.6891e-06	2.5134e-04	0.0018	0.0018	0.0024	0.1000
12	6.5196e-06	6.5196e-06	3.2254e-04	0.0019	0.0019	0.0026	0.1100
13	8.7752e-06	8.7752e-06	4.0321e-04	0.0021	0.0021	0.0029	0.1200
14	1.1498e-05	1.1498e-05	4.9321e-04	0.0023	0.0023	0.0031	0.1300
15	1.4728e-05	1.4728e-05	5.9229e-04	0.0025	0.0025	0.0033	0.1400
16	1.8504e-05	1.8504e-05	7.0018e-04	0.0026	0.0026	0.0036	0.1500
17	2.2864e-05	2.2864e-05	8.1655e-04	0.0028	0.0028	0.0038	0.1600
18	2.7845e-05	2.7845e-05	9.4107e-04	0.0030	0.0030	0.0040	0.1700
19	3.3481e-05	3.3481e-05	0.0011	0.0031	0.0031	0.0043	0.1800
20	3.9807e-05	3.9807e-05	0.0012	0.0033	0.0033	0.0045	0.1900
21	4.6855e-05	4.6855e-05	0.0014	0.0035	0.0035	0.0047	0.2000
22	5.4657e-05	5.4657e-05	0.0015	0.0037	0.0037	0.0050	0.2100
23	6.3242e-05	6.3242e-05	0.0017	0.0038	0.0038	0.0052	0.2200
24	7.2641e-05	7.2641e-05	0.0018	0.0040	0.0040	0.0055	0.2300

FIGURE 2.13: Sample data set

Based on the research in [60] and [61], the quadrotor can be sufficiently modelled by a second or third order system, the second order is used in this work, identified as ARX221:

$$A(z) = 1 - 1.8z^{-1} + 0.81z^{-2} \tag{2.54}$$

$$B(z) = 0.01z^{-2} \tag{2.55}$$

or, in the general sense:

$$A(z) = 1 + a_1z^{-1} + a_2z^{-2} \tag{2.56}$$

$$B(z) = b_1z^{-1} + b_2z^{-2} \tag{2.57}$$

where b_1 is very small, approximately equal to zero. Technically, selecting a very low model order will likely produce a model that does not fully describe the dynamics of the system, while high orders, though capable of describing the dynamics more closely, pose the problem of having an unnecessarily complex model.

This model must be optimised to improve accuracy of the identified model, if possible. The Response Optimisation tool is used, with initial parameters, as shown in Table 2.3 and a cost function of the form (2.58), using the Gradient Method to optimise a_1, a_2, b_1 and b_2.

$$V(t) = \sum_{i=1}^{t} e^2 \tag{2.58}$$

TABLE 2.3: Model parameters

Parameter	Identified value	Initial value	Optimised value
a1	-1.8	0	-1.801
a2	0.81	0	0.811
b1	0.01	0	0.01
b2	0	0	0

The optimisation is implemented as shown in Figure 2.14.

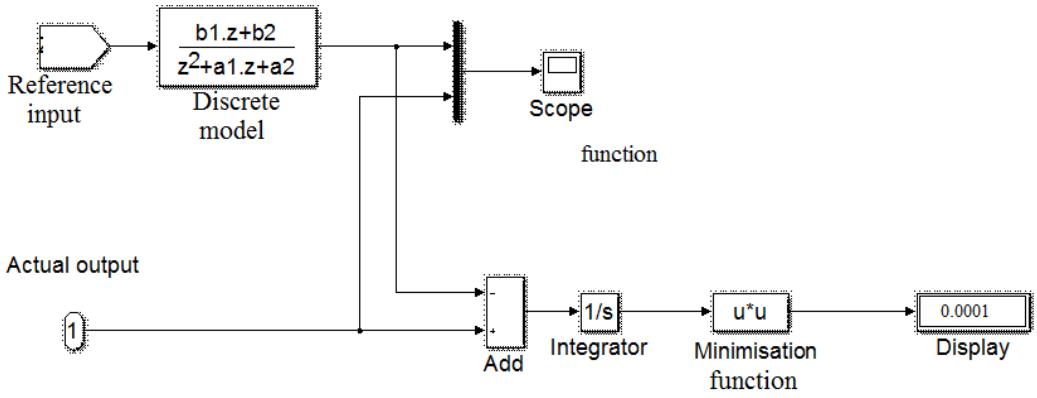

FIGURE 2.14: Optimisation scheme for identified model

Model validation

The final stage is model validation. The idea is to adopt the model identified from the step response data, run a ramp input simulation for this model then compare the resultant ramp response against the ramp response of the real system. Figure 2.15 illustrates the results.

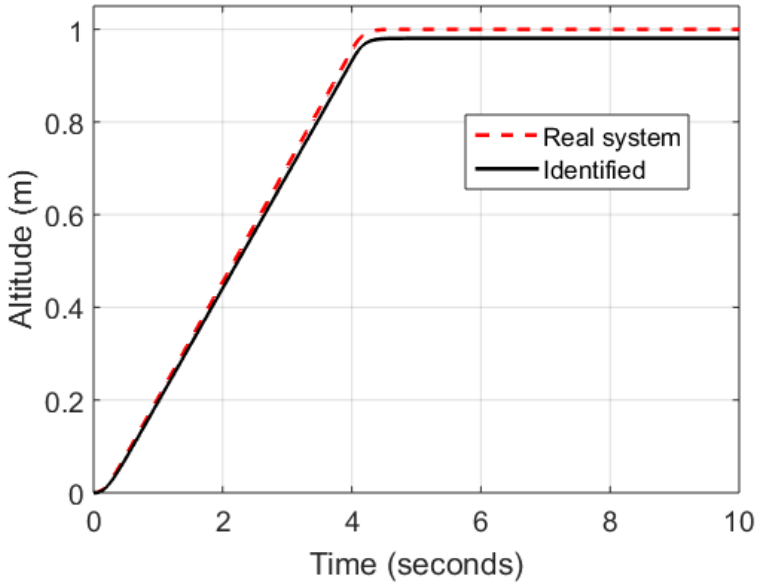

FIGURE 2.15: Evaluation by ramp response

From Figure 2.15, it is concluded that ARX221 is sufficient to describe the altitude system dynamics.

Further identification

The same model structure (2.56 and 2.57) is sufficient to model the lateral (y) and longitudinal (x) dynamics of the quadrotor too and a 3-dimensional plot for the PID trajectory tracking is shown in Figure 2.16.

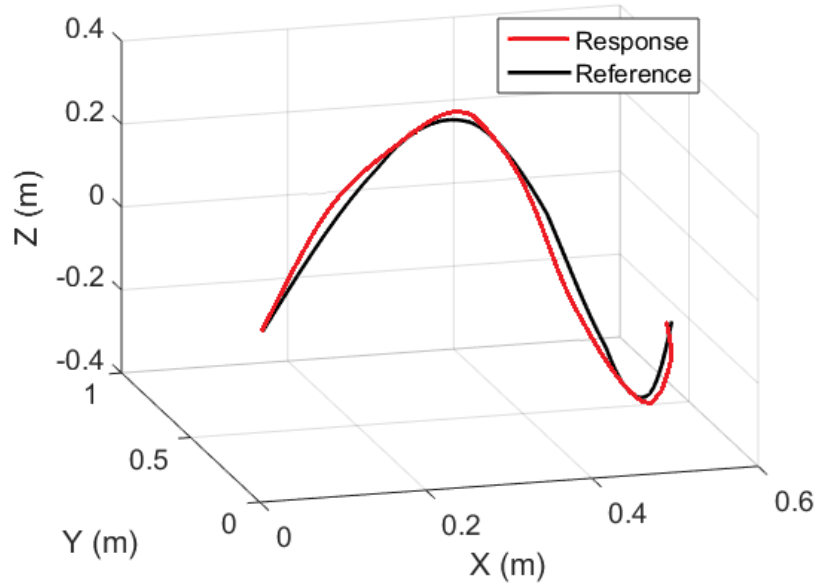

FIGURE 2.16: PID trajectory tracking

Following the same procedure for the flight experiment data, such as shown in Appendix A, yields comparable models as detailed in the next section.

2.2.3 Experimental approach

The experiments are conducted with a clear sense of safety, using the Axe quadrotor. The experiments are limited to a space with surface ground area not more than 8m*8m in length and width and maximum altitude of 12 meters. This is monitored and controlled by a 'Geo-fence' algorithm which comes with the autopilot (Pixhawk autopilot). Pixhawk has several flight modes, each with particular functions and characteristics. Some of the flight modes are:

1. Manual mode. Pure radio control, no position or attitude stabilization.
2. Loiter mode. Radio control with GPS assistance for position stabilization.
3. Auto mode. Autonomous GPS way-point tracking.
4. Altitude hold mode. This mode uses the barometric or ultrasonic sensor to stabilise altitude, position and attitude are purely radio controlled in this mode.

The loiter mode is used at this stage, and the quadrotor operates in closed-loop, as illustrated by Figure 2.17 while flight data is recorded and stored by the flight controller.

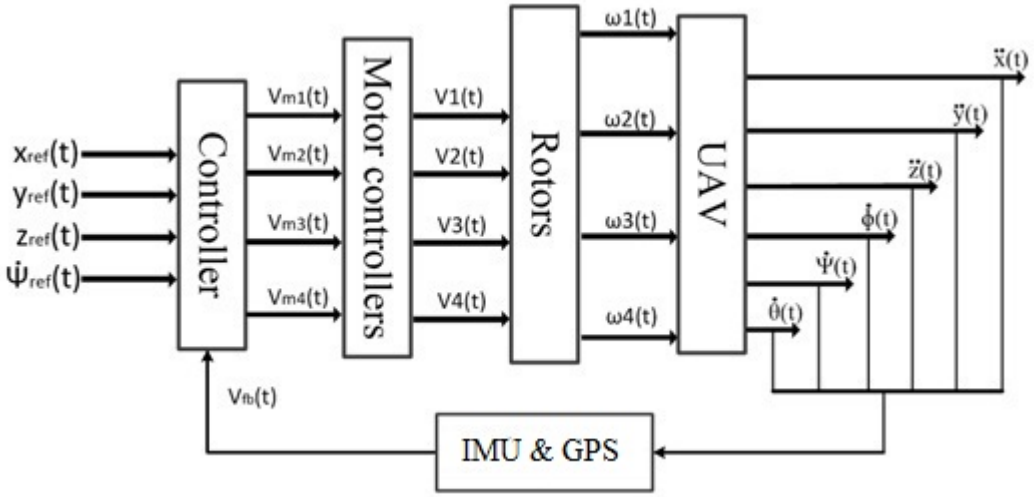

FIGURE 2.17: Closed-loop system

Figure 2.18, shows the experiment-ready physical system, the AXE Quadrotor. Modelling experiments are carried out on an outdoor field.

FIGURE 2.18: Experiment-ready physical system - AXE

In-built control and stability

The Axe quadrotor is a complete open-source system, with inbuilt but adjustable control and stability. The autopilot has been particularly tuned and optimised for the Axe frame at the level of manufacturing. The system realises PID control for all subsystems, which can be adjusted, but are already optimised.

Block diagram of the plant

The system diagram of the Axe quadrotor, showing all the main components and how they are set up is shown below in Figure 2.19.

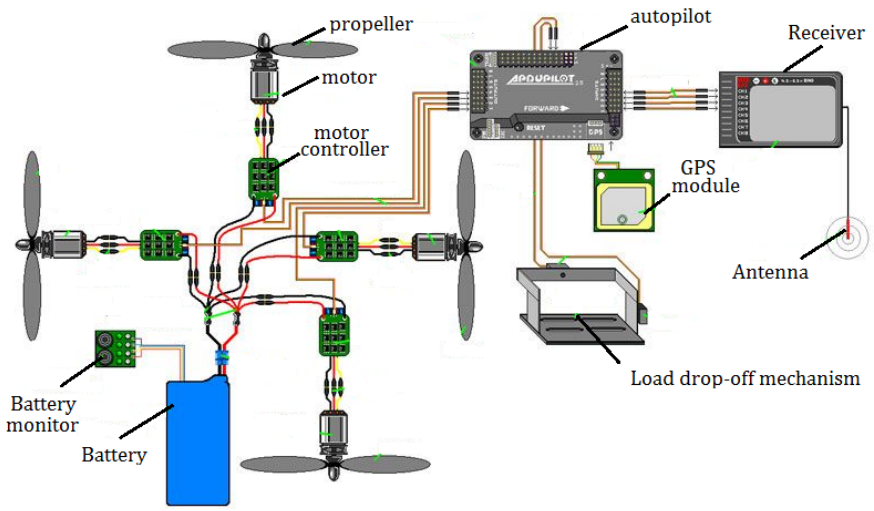

FIGURE 2.19: The system diagram of the Axe quadrotor

The block diagram of the Axe quadrotor is shown in Figure 2.20.

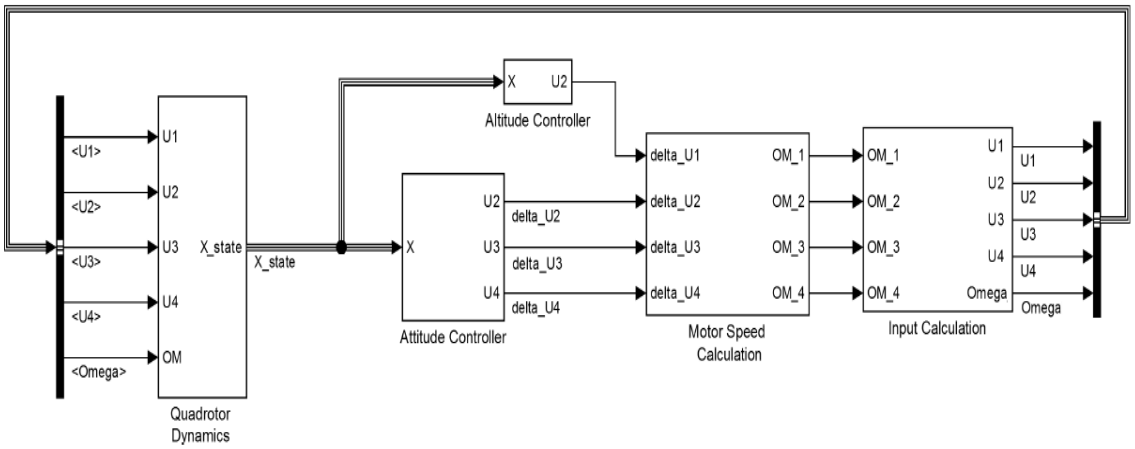

FIGURE 2.20: The block diagram of the Axe quadrotor

The actual plant detail is very similar to that already explained and illustrated earlier in this chapter. The data analysis software, Mission Planner, has an interface as illustrated by Figure 2.21 and Figure 2.22. These figures particularly demonstrate how different variables change in-flight, allowing determination of any interdependences.

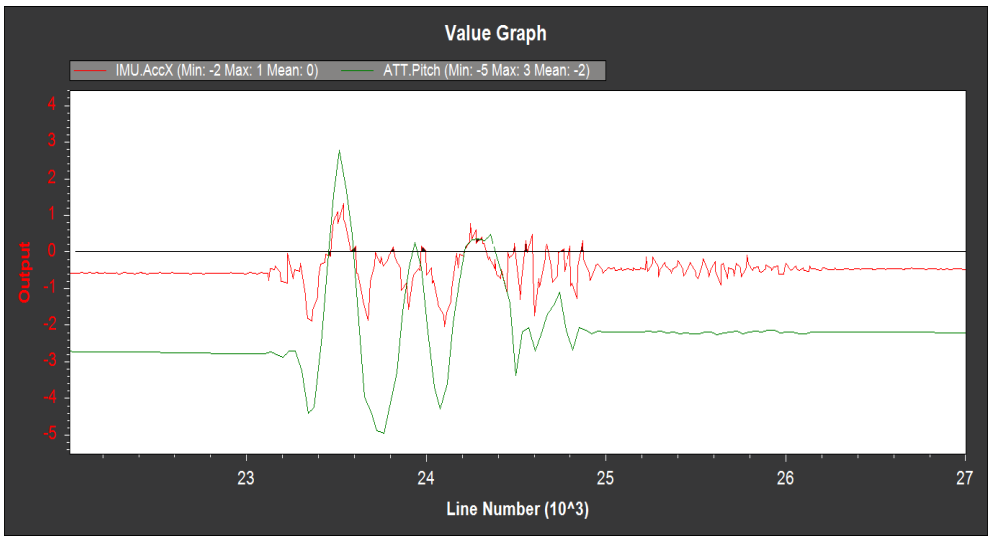

FIGURE 2.21: Flight log (1)

IMU.AccX (red plot) represents acceleration along the x axis and ATT.Pitch (green plot) represents pitch angle.

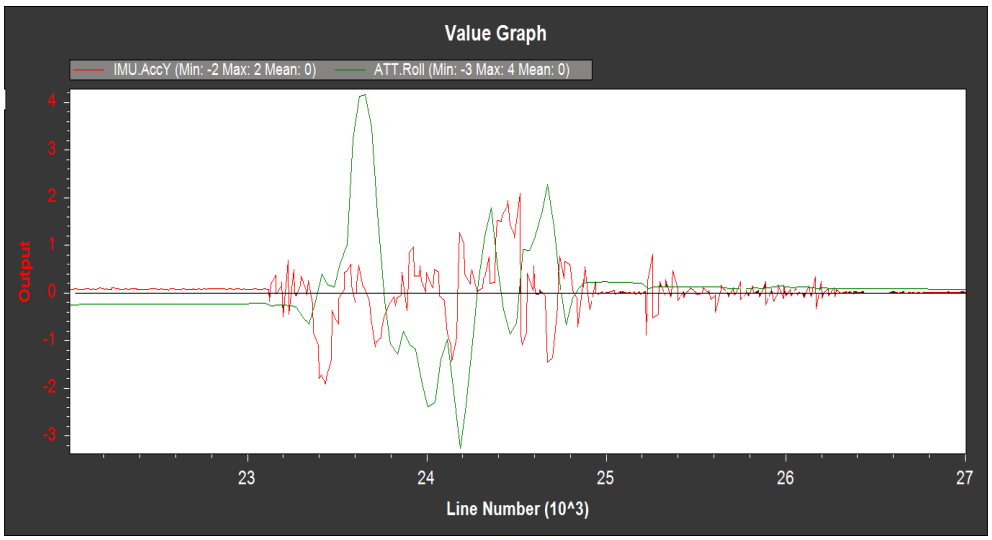

FIGURE 2.22: Flight log (2)

IMU.AccY (red plot) represents acceleration along the y axis and ATT.Roll (green) represents roll angle.

Now, following the same identification procedure as with analytical data, experimental data can also be processed. The resultant ARX structure, which is comparable to that obtained from the analytical-data system identification, is adopted. The actual coefficients of the model at this level are not important, only the structure.

According to (2.42) and (2.43) the acceleration along z, can be left out of this particular exercise as this is dependent on thrust, which cannot be measured by the system set up available. The same applies to ϕ, θ and ψ accelerations.

Validation

For model validation, first-order reduced models can be analysed. This validation is primarily based on the sign of the gain coefficients of the lateral and longitudinal control system identified models, in each case (depending on the orientation of the quadrotor). The gain must be of opposite signs for either system as is the case. More precise validation is possible, with some adjustments to the system, however, the model structure available at this point is sufficient for the objective of this work.

Graphical validation is also possible, where the experimental data can actually be plotted against the data from simulating the identified ARX system models. Figure 2.23 illustrates graphical validation, as for the altitude control system.

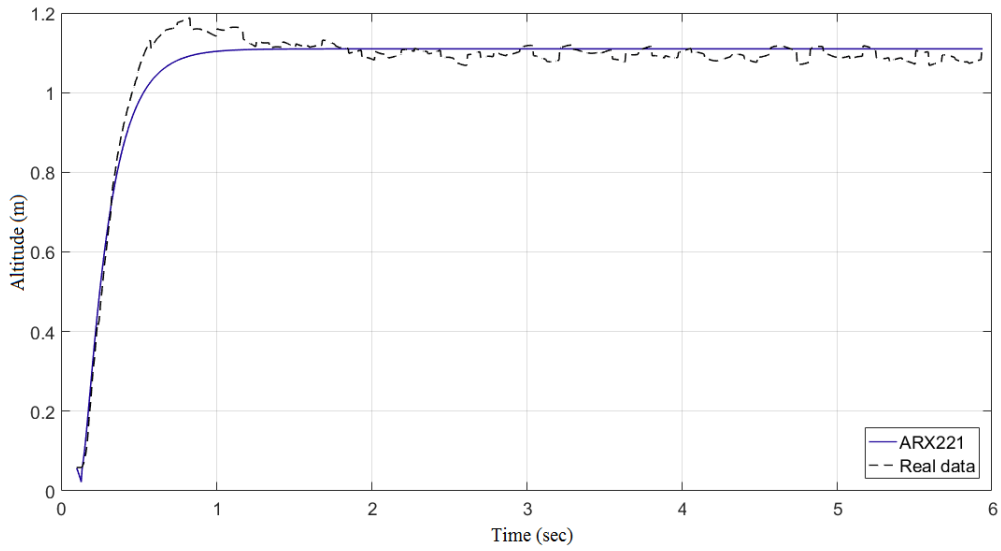

FIGURE 2.23: Graphical validation illustration

Some comparison indices are summarised in Table 2.4 and Figure 2.24 shows the cross-correlation for the input and output residuals.

TABLE 2.4: Graphical validation indices

Index	Value
Percentage fit	88.12
Steady-state error	< 0.015 m

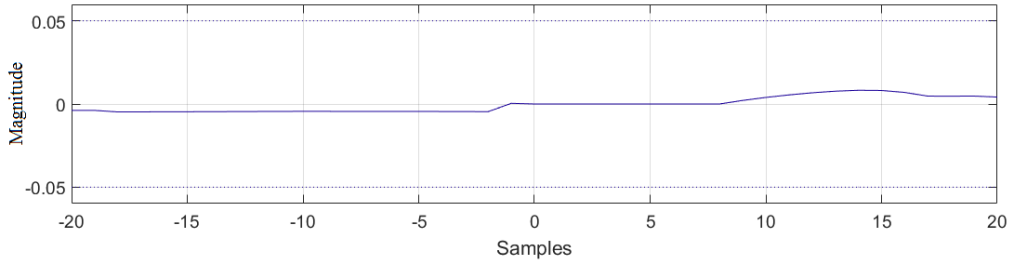

FIGURE 2.24: Cross-correlation for input and output residuals

Cross-correlation can be defined as a measure of the similarity of two signals, expressed as the displacement of one signal relative to the other.

2.3 Summary

The quadrotor is modelled analytically based on the Newton-Euler formalisation and experimentally from system identification based on simulation and experimental flight data. The identified model is in ARX form, a form selected because of its ease of operation with the procedures to be detailed in the next chapter (RLS parameter estimation procedures).

Chapter 3

AMPC DERIVATION

3.1 Adaptive control

Adaptive control is a technique that is based on automatic adjustment of control parameters in order to meet some desired performance index when the dynamics of the system are unknown and/or time-varying. This is especially useful in the case of the quadrotor intended to lift some unknown load and follow some desired trajectory. The minimum requirements for adaptive control are as follows:

1. A model structure to accurately represent the dynamics of the system.
2. A method for the estimation of the parameters of the model.
3. A suitable controller with adjustable parameters.
4. A mechanism to update the controller parameters.

3.1.1 Open-loop adaptive control

Adaptive control is highly flexible in terms of how this control technique can be classified. Open-loop adaptive control typically involves a look-up table type adaptation mechanism with measurements stored, of the corresponding controller parameters for a given data sample of measured environment conditions [5]. This therefore only works efficiently when a rigid relationship exists between the environmental conditions which are measured and the model parameters which are estimated. An example of this approach is gain-scheduling, a technique which has been successfully applied to autopilot systems.

3.1.2 Indirect adaptive control

Indirect adaptive control involves a design problem, which is solved based on the estimated system model parameters to obtain the controller parameters [5]. Indirect adaptive control therefore not only automates system modelling but control design too. Inaccuracy in estimated model parameters however, will typically result in very poor performance.

3.1.3 Direct adaptive control

Direct adaptive control involves direct updating of the controller parameters based on the estimated system model parameters. Direct adaptive control is generally more practical for implementation than indirect adaptive control. This research adopts a direct adaptive control scheme. The challenge is ensuring control signals generated are rich enough to expose the dynamics of system (i.e. meet the Excitation Condition). Figure 3.1 shows the general direct adaptive control scheme.

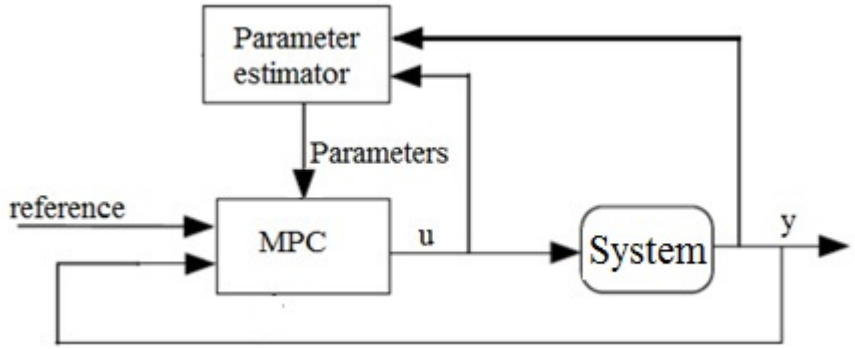

FIGURE 3.1: Direct adaptive control (adapted from [3] and [5])

The signal u represents control action and y represents the system output. Using these two signals, adaptive laws estimate parameters of the system model (within the parameter estimator block) in real-time, the model for which control is designed by updating the controller parameters.

3.1.4 Parameter estimation [3, 5, 6]

This research adopts the Recursive Least Squares (RLS) principle for parameter estimation. The RLS principle states that, the unknown parameters of a model must be chosen such that the sum of the squares of the difference between actual and computed values is minimum. Figure 3.2 shows more detail of the parameter estimator [3].

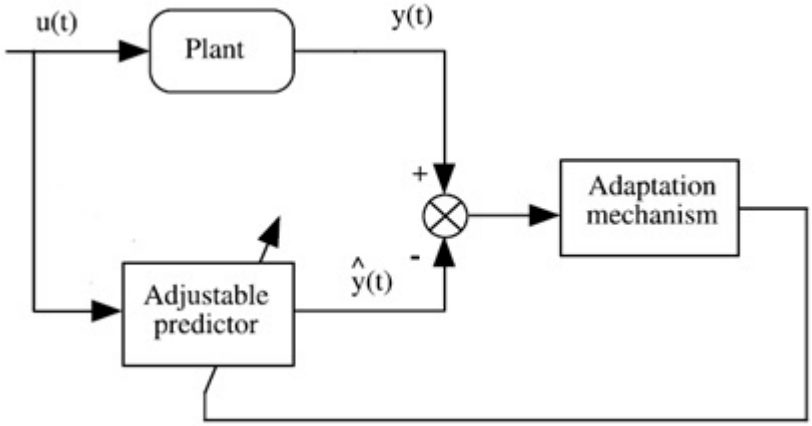

FIGURE 3.2: Detail of the parameter estimator [3]

$\hat{y}(t)$ is the model predicted output, based on the system model (with adjustable parameters) within the 'adjustable predictor' block. The error, between the system output at instant t and the output predicted by the model ($y(t) - \hat{y}(t)$), is processed within the 'adaptation mechanism' block based on some adaptive law, in this case, RLS. In this way, model parameters are updated at each sampling instant, minimizing the error. The output of the 'adaptation mechanism' are the estimated parameters. Derivations of adaptive control algorithms are found in textbooks such as [3], [5] and [6].

The LRS principle applied is based on a model represented in the Regressor Form as in equation (3.1) [3, 5, 6]

$$y(i) = \varphi_1(i)\theta_1 + \varphi_2(i)\theta_2 + ...\varphi_n(i)\theta_n \tag{3.1}$$

$y(i)$ is the measured output and time sample i. θ_1 to θ_n are the model parameters to be estimated while φ_1 to φ_n are some functions (can be linear or nonlinear). The equations that follow in this subsection, in detailing the RLS algorithm are referenced to [5] with further referencing to [3, 6] as well. Equation (3.1) can be written as (3.2)

$$\vec{y}(i) = \vec{\varphi}^T(i)\vec{\theta} \tag{3.2}$$

where $\vec{\varphi}^T$ and $\vec{\theta}$ are vectors and $\vec{\varphi}$ is particularly referred to as the Regressor. The RLS principle works efficiently with the regressor-type time function model and so it is necessary to reference the regressor-type model to the available model type. Assuming a deterministic environment and a SISO time-invariant system in the form of a discrete-time model:

$$y(t) = -\sum_{i=1}^{n_A} a_i y(t-i) + \sum_{i=1}^{n_B} b_i u(t-d-i) \tag{3.3}$$

where t is the normalised sampling time ($t = \frac{t}{T_s}$ and T_s = sampling period), $u(t)$ denotes input, $y(t)$ output, a_i and b_i are model parameters to be estimated and d is the number of sampling periods contained in the system time delay. n_A represents the number of time samples on the horizon of past outputs while n_B is the number of samples on the horizon of past inputs, delayed of course, by d samples. The system output is therefore, the sum of the weighted averages of past outputs over n_A and past inputs over n_B. Introducing the forward shift operator (q) or the backward shift operator (q^{-1}) results in a more convenient presentation of the model.

$$qy(t) = y(t+1) \tag{3.4}$$

$$q^{-1}y(t) = y(t-1) \tag{3.5}$$

Applying the shift operators gives

$$1 + \sum_{i=1}^{n_A} a_i q^{-i} = A(q^{-1}) = 1 + q^{-1}A^*(q^{-1}) \tag{3.6}$$

$$\sum_{i=1}^{n_B} b_i q^{-i} = B(q^{-1}) = q^{-1}B^*(q^{-1}) \tag{3.7}$$

where:

$$A(q^{-1}) = 1 + a_1 q^{-1} + \cdots + a_{n_A} q^{-n_A} \tag{3.8}$$

$$A^*(q^{-1}) = a_1 + a_2 q^{-1} + \cdots + a_{n_A} q^{-n_A+1} \tag{3.9}$$

$$B(q^{-1}) = b_1 q^{-1} + \cdots + b_{n_B} q^{-n_B} \tag{3.10}$$

$$B^*(q^{-1}) = b_1 + b_2 q^{-1} + \cdots + b_{n_B} q^{-n_B+1} \tag{3.11}$$

The model (3.3) can therefore be written as:

$$A(q^{-1})y(t) = q^{-d}B(q^{-1})u(t) = q^{-d-1}B^*(q^{-1})u(t) \tag{3.12}$$

then in forward time also as:

$$A(q^{-1})y(t+d) = B(q^{-1})u(t) \tag{3.13}$$

and also as:

$$y(t+1) = -A^*y(t) + q^{-d}B^*u(t) = -A^*y(t) + B^*u(t-d) \tag{3.14}$$

which can actually be represented in regressor form as in (3.15)

$$y(t+1) = \vec{\varphi}^T(t)\vec{\theta} \tag{3.15}$$

Further, if the elements on both sides of equation (3.12) where passed through a filter $\frac{1}{A(q^{-1})}$ the result would be the traditional:

$$y(t) = \frac{q^{-d}B(q^{-1})}{A(q^{-1})}u(t) = G(q^{-1})u(t) \tag{3.16}$$

with $G(q^{-1})$ being the transfer operator.

3.1.5 Task

The task at hand is therefore to determine $\vec{\theta}$ such that the output from the estimated model $\hat{y}(i)$ matches as closely as possible to measured/actual output $\vec{y}(i)$. This means that error ε, as in equation (3.17), must be minimal.

$$\varepsilon(i) = y(i) - \hat{y}(i) \tag{3.17}$$

A quadratic cost function (V) is introduced as in equations (3.19) and (3.18) [3]

$$V(\vec{\theta}, t) = \frac{1}{2}\sum_{i=1}^{t}[\vec{y}(i) - \hat{y}]^2 \tag{3.18}$$

$$V(\vec{\theta}, t) = \frac{1}{2}\sum_{i=1}^{t}[\vec{y}(i) - \vec{\varphi}^T(i)\hat{\theta}]^2 \tag{3.19}$$

The global minimum of the cost function (3.20) corresponds to the derivative of the cost function with respect to $\vec{\theta}$ equated to zero.

$$\frac{\partial V}{\partial \vec{\theta}} = (\vec{y}(i) - \vec{\varphi}^T(i)\hat{\theta})(-\vec{\varphi}^T(i)) = 0 \tag{3.20}$$

It is therefore possible to solve for $\vec{\theta}$, which corresponds to the estimated parameters $\hat{\theta}$, in an equation (3.21) referred to as the Least Squares Formula

$$\hat{\theta} = (\vec{\varphi}^T\vec{\varphi})^{-1}\vec{\varphi}^T\vec{y} \tag{3.21}$$

The matrix $\varphi^T\varphi$ is always non-negative definite and so cost function V has a unique minimum. Proof of this formula is available in [3, 5, 6].

Adaptive Gain

For better presentation and even exploration, a variable P can be introduced according to equations (3.22) and (3.23) [3]

$$P(t) = (\vec{\varphi}^T(t)\vec{\varphi}(t))^{-1} = \sum_{i=1}^{t}\vec{\varphi}^T(i)\vec{\varphi}(i))^{-1} \tag{3.22}$$

$$\hat{\theta}(t) = P(t)(\sum_{i=1}^{t}(\vec{\varphi}^T(i)\vec{y}(i)) \tag{3.23}$$

P is referred to as the adaptive gain as will be detailed later in this chapter. In [3], P is clearly defined as some measurement of the progression of the covariance of the error in parameter estimation. The matrix $\vec{\varphi}^T\vec{\varphi}$ must be invertible, a condition known as the excitation condition [5]. If this condition is met, it means that input signals to the system are rich enough to characterise the dynamics of the system.

3.1.6 A more practical approach

For practical implementation, it is necessary to consider a model (discrete-time model) which makes use of 'priory predicted output' [3, 6] as presented in equation (3.24)

$$y(t+1) = a_1\varphi_1(t) + b_1\varphi_2(t) = \vec{\theta}^T\vec{\varphi}(t) \tag{3.24}$$

where

$$\vec{\theta} = [a_1, b_1] \tag{3.25}$$

$$\vec{\varphi}(t) = [\varphi_1(t), \varphi_2(t)] \tag{3.26}$$

The prediction model can therefore be written as (3.27)

$$\hat{y}(t+1) = \hat{a_1}(t)\varphi_1(t) + \hat{b_1}(t)\varphi_2(t) = \hat{\theta}^T\vec{\varphi}(t) \tag{3.27}$$

This means that the error minimised is in fact $\varepsilon^2(t+1)$ rather than $\varepsilon^2(t)$. The cost function to be minimised can be represented as follows [3, 5, 6]:

$$V(\vec{\theta}, t) = \frac{1}{2}\sum_{i=1}^{t}[\vec{y}(i) - \hat{\theta}^T(t)\vec{\varphi}(i-1)]^2 \tag{3.28}$$

This makes more sense in that, the predicted output at the time instant $i(i \leq t)$ is based on the estimated parameters at the time instant t which are obtained using t measurements. The parameter θ must be estimated at the time instant t and must be such that the sum of the squares of the differences between the system output and the prediction model output, within a horizon of t samples, is minimised. This is done by partial derivation of the cost function [fully detailed in 5] with respect to $\vec{\theta}$ and results in:

$$\hat{\theta}(t) = P(t)\sum_{i=1}^{t} y(i)\varphi(i-1) \tag{3.29}$$

where

$$P(t)^{-1} = \sum_{i=1}^{t}\varphi(i-1)\varphi^T(i-1) \tag{3.30}$$

Equation (3.29) represents the parameter estimator law used in estimating parameters of the system.

3.1.7 RLS parameter estimation algorithm

The idea behind recursion is to use results obtained at time sample t to compute estimates at time sample $t+1$. In general, the estimated parameter output is equal to the previous value of estimated parameters plus some correcting term which depends on the most recent measurements [3].

$$\begin{bmatrix} new \\ estimated \\ parameters \\ (vector) \end{bmatrix} = \begin{bmatrix} previous \\ estimated \\ parameters \\ (vector) \end{bmatrix} + \begin{bmatrix} adaptation \\ gain \\ (matrix) \end{bmatrix} \times \begin{bmatrix} measurement \\ function \\ (vector) \end{bmatrix} \times \begin{bmatrix} functionof \\ error \\ (scalar \end{bmatrix} \tag{3.31}$$

To factor in recursion, equation (3.29) is manipulated to suit the expression above. This mathematical manipulation is explained step by step in [3]. The final recursive representation is as follows:

$$\hat{\theta}(t+1) = \hat{\theta}(t) + P(t+1)\varphi(t)\varepsilon(t+1) \tag{3.32}$$

where

$$\varepsilon(t+1) = y(t+1) - \hat{\theta}^T(t)\varphi(t) \tag{3.33}$$

The part $P(t+1)$ in equation (3.32) is the gain matrix of the adaptive law as already stated. This gain matrix must be adjusted with recursion such that while in the vicinity of the minimum, smaller steps in gradient are taken (to avoid oscillation) but larger steps are taken while far from the minimum (to achieve high convergence speed) [5]. The expression for P however, is in terms of $P(t+1)^{-1}$ instead of $P(t+1)$ which is the form in the estimator. This can be solved by using the Matrix Inversion lemma, as detailed in [3, 5, 6], and completes the RLS law with:

$$P(t+1) = P(t) - \frac{P(t)\varphi(t)\varphi^T(t)P(t)}{1+\varphi^T(t)P(t)\varphi(t)} \tag{3.34}$$

The RLS parameter estimation algorithm is summarised by the equations (3.32), (3.33)and (3.34).

The RLS algorithm must start from an initial estimation given at time point t_0 where $t_0 = dim\varphi(t)$, basing on the fact that $P(t)^{-1}$ would typically become non-singular for $t < t_0$ [3]. Practically, the algorithm is initiated by selecting:

$$P(0) = \frac{1}{\delta}I = (GI)I \tag{3.35}$$

where $0 < \delta \ll 1$ and with typical values $\delta = 0.001$ i.e. $GI = 1000$. Analysing the RLS algorithm, the influence of the initial error is observed to decrease in time, and

the criterion to be minimised is as follows [3]:

$$\min_{\hat{\theta}(t)} V(t) = \sum_{i=1}^{t} [y(i) - \hat{\theta}^T(t)\varphi(i-1)]^2 + [\theta - \hat{\theta}^T(0)]^T P(0)^{-1} [\theta - \hat{\theta}(0)]^T \tag{3.36}$$

Given $P(0) > 0$, for the matrix $P(0)$:

$$\lim_{t \to \infty} \varepsilon(t+1) = 0 \tag{3.37}$$

Considering single parameter estimation where $P(t)$ and $\varphi(t)$ are scalars, it is possible to further determine whether or not the adaptation gain of the RLS algorithm is of a decreasing type or not. For this condition, the expression for $P(t+1)$ becomes:

$$P(t+1) = \frac{P(t)}{1 + \varphi(t)^2 P(t)} \leq P(t) \tag{3.38}$$

where $\varphi(t), P(t) \epsilon R^1$. It can be observed that the adaptation gain of the RLS algorithm is of a decreasing type. Moreover, $\varphi(t)\varphi^T(t)$ being greater or equal to zero, it can be concluded that if $\varphi(t)\varphi^T(t)$ is greater than zero in average, then $P(t)^{-1}$ tends to infinity and so $P(t)$ tends towards zero.

The RLS algorithm can be generalised for any discrete-time model (of any dimension) of the form:

$$y(t) = \frac{q^d B(q^{-1})}{A(q^{-1})} u(t) \tag{3.39}$$

which, as shown earlier in (3.1.4) can be written in the form:

$$y(t+1) = -\sum_{i=1}^{n_A} a_i y(t+1-i) + \sum_{i=1}^{n_B} b_i u(t-d+1-i) = \theta^T \varphi(t) \tag{3.40}$$

where:

$$\theta^T = [a_1, \ldots, a_{n_A}, b_1, \ldots, b_{n_B}] \tag{3.41}$$

$$\varphi^T(t) = [-y(t), \ldots, -y(t-n_A+1), u(t-d), \ldots, u(t-d-n_B+1)] \tag{3.42}$$

The general predictions are given by:

$$\begin{array}{rcl} \hat{y}(t+1) & = & -\sum_{i=1}^{n_A} \hat{a}_1(t) y(t+1-i) + \sum_{i=1}^{n_B} \hat{b}_1 u(t-d+1-i) \\ & = & \hat{\theta}^T(t)\varphi(t) \end{array} \tag{3.43}$$

where:

$$\hat{\theta}^T(t) = [\hat{a}_1(t), \ldots, \hat{a}_{n_A}(t), \hat{b}_1(t), \ldots, \hat{b}_{n_B}(t)] \tag{3.44}$$

the estimation of whose parameters can be done by the RLS algorithm already detailed in this chapter. The RLS algorithm discussed thus far applies to time-invariant systems, on the basis that less and less weight is given to new measurements, as to new prediction errors. The parameters of the quadrotor model are however, time

varying. There are several methods to adjust the RLS algorithm for time-varying models including using:

- Forgetting Factor
- Constant Trace
- Decreasing (Vanishing) Gain
- Kalman Filter
- Constant Gain (Gradient Algorithm)

These different methods work by means of manipulation of the adaptation gain and can be used in combinations. Each approach has specific advantages depending on the nature of the problem to be solved, but as a basis [5], the Forgetting Factor approach can be tuned to suit both systems with time-invariant and time-variant parameters. Constant Trace can handle systems with rapid time-varying parameters while the Decreasing Gain algorithm is recommended for stationary systems. The Gradient Algorithm, while simpler to implement, only works efficiently with systems with few parameters, typical a maximum of three and in the presence of low noise. The Kalman Filter algorithm is relatively more flexible, in the sense that, the algorithm can be tuned to suit time-invariant parameters, slowly varying parameters or even rapidly varying parameters.

3.1.8 Manipulating the adaptive gain

The inverse adaptation gain can have weights $\lambda_1(t)$ and $\lambda_2(t)$ introduced as [5]:

$$P(t+1)^{-1} = \lambda_1(t)P(t)^{-1} + \lambda_2(t)\varphi(t)\varphi^T(t) \tag{3.45}$$

where $\lambda_1(t)$ is greater than zero and less than or equal to one, $\lambda_2(t)$ is greater than or equal to zero and less than 2 and $F(0)$ is greater than zero. $\lambda_1(t)$ and $\lambda_2(t)$ work in 'opposite' in the sense that $\lambda_1(t) < 1$ tends to increase adaptation gain while $\lambda_2(t)$ decreases the adaptation gain. Depending on the selection of $\lambda_1(t)$ and $\lambda_2(t)$ values, the parameter estimation algorithm can be controlled to behave in a particular manner according to the requirements at hand.

Forgetting Factor

In the case of Constant Forgetting Factor [5]:

$$\lambda_1(t) = \lambda_1; \lambda_2(t) = \lambda_2 = 1 \tag{3.46}$$

where λ_1 greater than zero but less than one. λ_1 being less than one means that the weighting weakens on older data, maximising weight on the latest data.

In the case of Variable Forgetting Factor:

$$\lambda_2(t) = \lambda_2 = 1 \tag{3.47}$$

and

$$\lambda_1(t) = \lambda_0\lambda_1(t-1) + 1 - \lambda_0 \tag{3.48}$$

where λ_0 (initial condition) greater than zero but less than one. This setup will hold high gain for regions relatively far away from the optimum and will lower gain as the optimum approaches, consequently speeding up convergence. A more practical realisation of the Forgetting Factor can be considered if weighting is applied to the cost function (3.19) to give:

$$V(\hat{\theta}, k) = \frac{1}{2}\sum_{i=1}^{k} \lambda^{k-i}[y(k) - \varphi^T(k)\hat{\theta}(k)]^2 \tag{3.49}$$

which is referred to as least squares with exponential forgetting. Smaller values of λ result in a quicker loss (forget) of the previous data giving the solution [3]:

$$\hat{\theta}(t) = \left[\sum_{k=1}^{t} \lambda^{k-i}\varphi(k)\varphi^T(k)\right]^{-1} \sum_{k=1}^{t} \lambda^{k-i}y(k)\varphi(k) = P(t)\sum_{k=1}^{t} \lambda^{k-i}y(k)\varphi(k) \tag{3.50}$$

where:

$$P(t) = \sum_{k=1}^{t} \left(\lambda^{k-i}\vec{\varphi}^T(k)\vec{\varphi}(k)\right)^{-1} \tag{3.51}$$

The idea is that when λ is close to 1, the result is good convergence and small variances of the estimates.

3.1.9 Adaptive control implementation

Adaptive control design is simulation intense. The idea is to find a starting point, then gradually improve the controller according to simulation results and background knowledge. Figure 3.3 illustrates an implementation scheme for online estimation of a process.

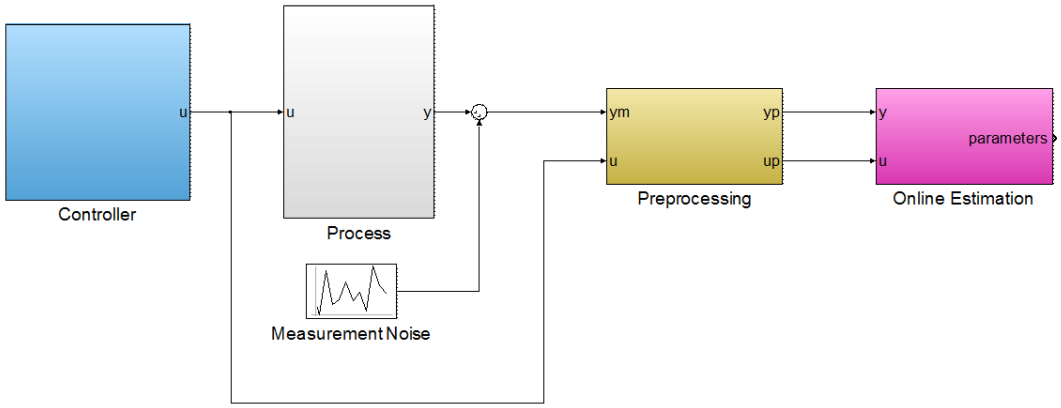

FIGURE 3.3: A scheme for online estimation

The preprocessing block is meant to refine data passed into the estimator and typically includes removal of outliers or sensor bias etc. The On-line Estimation block is where parameter estimation takes place. The estimator can be implemented by Matlab file code or the Matlab toolbox, the System Identification Toolbox.

Persistent excitation

The quality of the estimated parameters is affected by the properties of the signals (system inputs) used by the parameter estimator. Persistent excitation refers to the provision of input signals to the system, with characteristics that enables comprehensive display of the system dynamics [5].

Persistent excitation typically guarantees exponential convergence of the estimated variables to the optima. This topic has several ways to look at it and many papers present derivations of the general expressions that can be used when analysing this aspect of persistent excitation [3, 5, 70, 71], including the general theorem. It is important to note that methods exist, to achieve exponential convergence without the need to meet the 'restrictive' (as referred to in [72]) persistent excitation condition [72, 73], and typically, stochastic noise signals allow persistent excitation. The approach taken in this work involves inclusion of an extra perturbation signal to the system, to allow persistent excitation. This signal acts on the closed-loop system before calculation of the error, in each subsystem case. Figure 3.4 illustrates the implementation where a white stochastic noise signal generator is utilised to provide the perturbation signal.

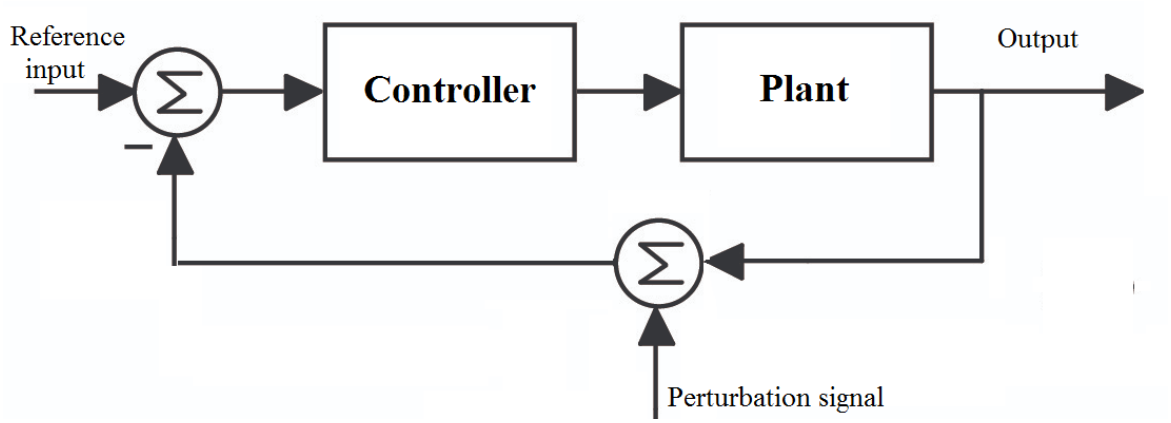

FIGURE 3.4: Persistent excitation realisation

Several approaches can be taken when developing the adaptive controller. Ultimately, the controller must satisfy the desired stability and performance requirements.

Starting Values

The initial model parameters are $\hat{\theta}(0)$. $P(0)$ represents some estimate of the covariance matrix of the initial parameters. Small $P(0)$ results in small $K(t)$ and consequently small changes of the estimated parameters $\hat{\theta}(t)$. On the other hand, large $P(0)$ results in quick a rapid move from $\hat{\theta}(0)$ to $\hat{\theta}(t)$ [74]. Typically, for the practical case:

$$\hat{\theta}(0) = 0 \tag{3.52}$$

and

$$P(0) = \rho \mathbf{I} \tag{3.53}$$

where ρ is a gain constant and $\mathbf{I}$ is a unitary identity matrix. It makes sense therefore, to use a large ρ when the initial estimates are uncertain, in which case, low confidence is given to $\hat{\theta}(0)$ and then rapid transient until the final parameter values are met. Reducing the value of ρ means that the gain factor is reduced which then results in slower convergence. The effects of the tuning parameters of the parameter estimator are investigated next.

Effects of the Forgetting Factor

The effect of the Forgetting Factor, as explained earlier, is that a decrease in the value of λ results in [74] the parameter estimates reaching the true value quicker.

The forgetting factor directly affects the several indices of the RLS algorithm including, and especially stability and convergence [75]. To investigate the effects of the forgetting factor, a Recursive Least Squares Estimator is run from m-file code.

Running this simulator allows comparison of the response of the modelled system against the response of the estimated model.

The estimator is developed for analysis purposes. λ typically takes values between 0.95 and 1 [3, 75] and it is common practice to select staring parameter values as zero while the covariance matrix of parameters typically takes a wide range of values.

The strategy implemented in tuning the estimator is as follows:

1. Tune and set λ for $\hat{\theta}(0) = 0$.
2. Select values for the covariance matrix, with initial $\rho = 100$.

In tuning λ, the estimator is first run for the case without forgetting (λ = 1). Figure 3.5 shows a comparison of the response of the estimated system versus the true response, for different λ settings.

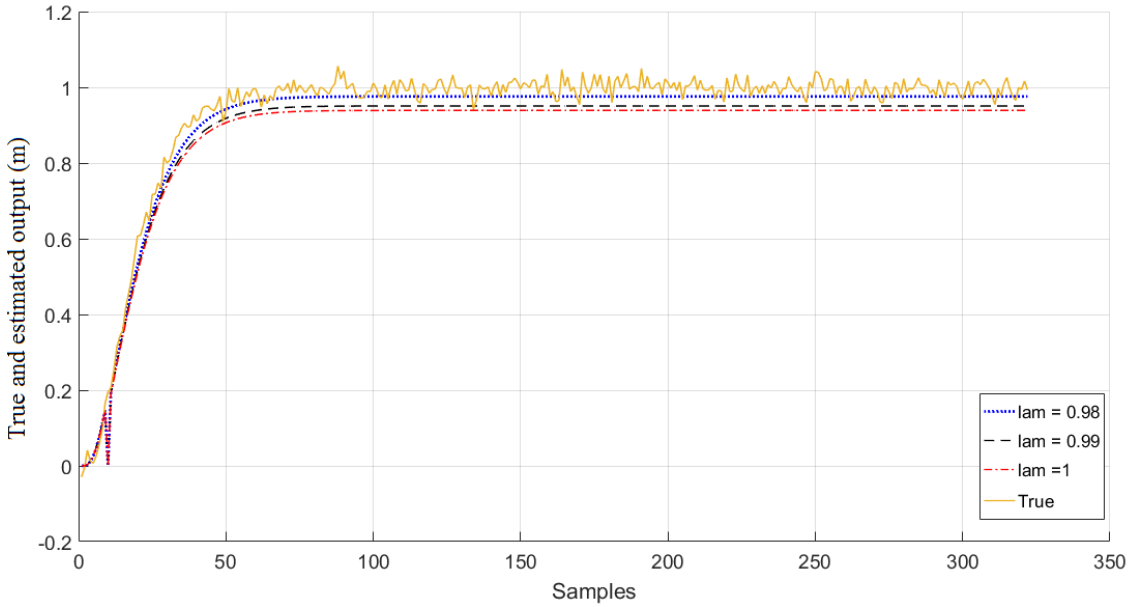

FIGURE 3.5: Effects of λ

Analysis of several outcomes reveals that the forgetting factor (λ), has some effect on the convergence of the estimated model response. Reducing λ results in more reliable model estimation where steady state error lessens while convergence becomes more rapid. The value settled for is $\lambda = 0.98$. This means that the estimated quadrotor model parameters become more reliable as λ decreases from 1 to approximately 0.98. This should mean more accurate capture of the quadrotor dynamics.

Effects of the gain constant

The ρ has an effect on the convergence and on the rate of convergence of parameters. Increasing ρ, as expected, not only brings satisfactory convergence accuracy, but also increases the rapidness of estimated model response in matching the true response.

Figure 3.5 shows a comparison of the response of the estimated system versus the true response, for different ρ settings.

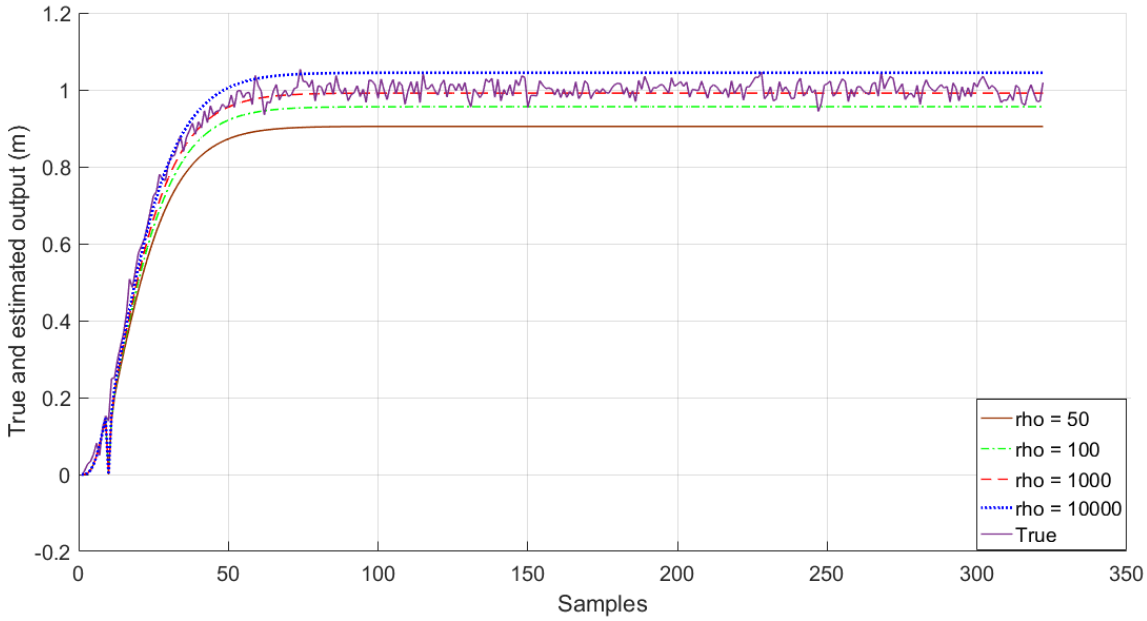

FIGURE 3.6: Effects of ρ

This means that the estimated quadrotor model parameters converge more rapidly as ρ increases. The setting of $\rho = 1000$ and $\lambda = 0.98$ provides such rapid convergence and at the same time, high accuracy of the estimated model response to the real response. Values of ρ that are too high however do increase initial convergence rapidness but at the cost of poor recovery, should the true response change, say for example from transient to steady state. Further analysis of the adaptive controller follows towards the end of the chapter, on the complete system which includes the full controller, as shown in Figure 3.7.

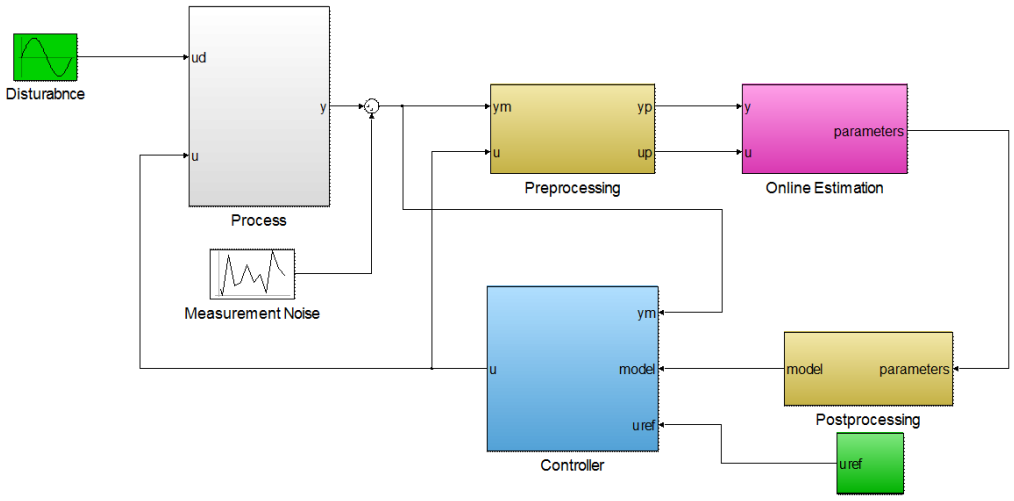

FIGURE 3.7: General adaptive controller

In this way, $uref$ is the system reference input, in this case for example reference longitudinal trajectory. ud is the disturbance affecting the system, which can be measured and possibly fed-forward to enhance control design, will be explored later. This disturbance, for the quadrotor, will simulate the varying load effects. The whole idea is therefore to design a controller that can make corrections for the disturbance. The 'Process' block represents the plant, that is, the quadrotor. This plant has some output y which is measured by on-board sensors. To make the measurement more realistic, some noise can be added to the output signal through the 'measurement noise' block. The 'post processing' block is typically used for conversion of the model type into a suitable model (for example ARX to state space), deployed in the 'controller'. Based on the up-to-date model, the controller can then compute control action (manipulated variable) u in accordance to the real-time system dynamics.

3.2 Model predictive control

Model Predictive Control (MPC) has seen significant development over the past few decades. MPC is an optimal control technique employing an internal model to predict system behaviour in some finite time horizon. As such, MPC is also referred to as Moving Horizon Control (MHC) or Receding Horizon Control (RHC). Using the model predictions, it is possible to optimise some performance index by generating a sequence of control action into the predicted time horizon.

MPC gained early popularity in control of slow dynamical systems, especially in chemical process control, mainly petrochemical process control [76]. Recent advances in processing power have seen MPC spreading into fields of control where system dynamics are fast, such as the type considered in this research, i.e. quadrotor control system. The idea behind MPC is to minimise the difference between the model predicted output and the desired reference. This difference is minimized over some time horizon with finite time samples. The process can be directly subjected to constraints on the control action, outputs and states. MPC is based on the receding horizon philosophy which can be summarised as follows:

At time sample t, from the sequence of calculated control action commands, only the first input is applied to the plant and the rest deleted. The optimal problem is solved again at time sample $t + 1$ and a new sequence of control action calculated.

Common forms of MPC are DMC, Model Algorithm Control (MAC), Predictive Functional Control (PFC) and Generalised Predictive Control (GPC). DMC and GPC are more common. The differences are rather subtle, but important to note with DMC and GPC is that DMC, which was developed and introduced by Charles Cutler in the 1970s [77, 105], uses a stable step response model, and puts relatively more emphasis on past input data to make predictions [78]. DMC is less sensitive to noise because it is non-parametric (utilising a step response model with 60 or 100 data points for example), although a little more computationally expensive than GPC. MAC utilises an impulse response, again, only applicable to stable systems. MAC however is typically even more computationally expensive [79]. PFC, developed by

Rachalet [80] uses state space models, and is generally more complex than DMC or MAC. GPC utilises Controlled Auto Regressive Moving Average (CARIMA) models [81] and as stated, is relatively more sensitive to noise. It should be noted though that, methods do exist, to make GPC less sensitive to noise (for example implementing filters). For the advantage of noise sensitivity, the type of MPC investigated in this work is DMC, leading to Adaptive Dynamic Matrix Control (ADMC).

3.2.1 Structure of MPC

MPC utilises two main components, i.e. the model and the optimisation solver. Figure 3.8 details the structure of an MPC scheme. The basics of MPC can be summarised as follows:

- Model prediction of process output in the prediction horizon.
- The receding horizon technique, i.e. forward shift of the prediction horizon into the future at each time sample.

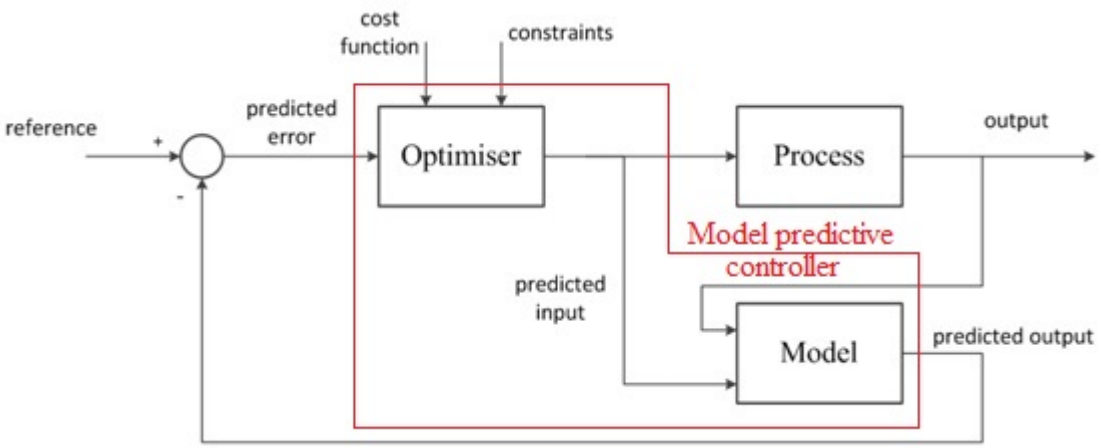

FIGURE 3.8: Basic MPC scheme - adapted from [82]

Derivations of various MPC algorithms are presented in [82] and [83]. MPC is based on the internal model deployed. Process models are commonly used in MPC, which can be:

- state space models (used mostly in research)
- transfer function models
- impulse response models (mostly industrial)
- step response models

Disturbance models can also be used, commonly the Controlled AutoRegressive and Integrated Moving Average (CARIMA) models. Some of the aspects that have fuelled the popularity of MPC are that MPC:

- offers optimal tracking (with compensation of model uncertainty).
- allows direct incorporation of constraints to represent limitations of certain parts of a system for example actuator limits.
- is able to handle the multi-variable quadrotor system.
- can handle non-minimal phase and unstable processes.
- is flexible in terms of tuning.

MPC can be classified according to linearity (linear and non-linear MPC). Linear MPC will be used in this research. The methodology of MPC comes down to three main steps, step 1 to step 3 [83], illustrated by Figure 3.9.

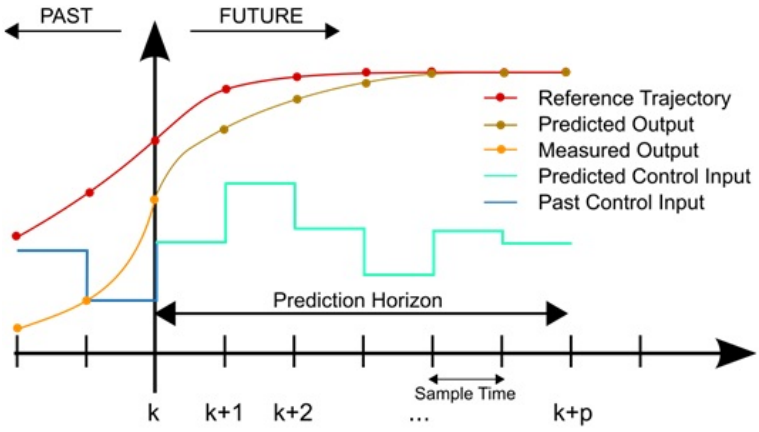

FIGURE 3.9: The methodology of MPC [86]

Step 1: The predicted outputs are denoted by $\hat{y}(t+k|t)$, where $k = 1, ..., N$, (N being the prediction horizon) and the t preceded by the vertical slash is the point in time, at which the prediction is made. The predicted outputs are calculated at each time sample from the system model. Contributing to this calculation, are the past inputs and outputs, the current output (denoted $y(t)$) and the future control signals to be calculated, $u(t+k|t)$, where $k = 0, ..., N-1$.

Step 2: The sequence of control signals required to minimise the error (difference between desired and predicted trajectories) is calculated.

Step 3: Only the current control signal $u(t|t)$ is applied to the system. For the next sampling time, $y(t+1)$ is measured and step 1 is repeated. In this way, the prediction horizon translates by a sampling interval at each computation.

3.2.2 Cost function

Equation (3.54) represents the general cost function [82, 83].

$$J = \sum_{i=1}^{p} (y^{ref}(k+i) - \hat{y}(k+i))^2 + r \sum_{i=0}^{c-1} \Delta u(k+i)^2 \tag{3.54}$$

where

- $k \epsilon \mathbf{Z}$ is the sampling instant and the real time t, relates to k by $t = kT_s$ where T_s is the sampling time.
- y^{ref} is the reference trajectory.
- y^{sp} is the set-point.
- $\hat{y}(k)$ is the predicted output at instant k.
- $\Delta u(k)$ is the control move at instant k and $\Delta u(k) = u(k) - u(k-1)$.
- p is the prediction horizon, c is the control horizon.
- r is a weighting factor and generally represents the ratio of importance between costs due to deviation from desired output and costs due to control moves.

To minimise the cost function, an optimisation problem must be solved, which can be constrained in the system output ($y_{min} < y < y_{max}$), manipulated variable ($u_{min} < u < u_{max}$) and change of manipulated variable ($\Delta u_{min} < \Delta u < \Delta u_{max}$).

y^{ref}, which is typically an exponential rise, is the reference trajectory to move from measured output to the set-point. The expression for y^{ref} can be manipulated to tune the rate at which the output approaches the set-point, as shown in Figure 3.10.

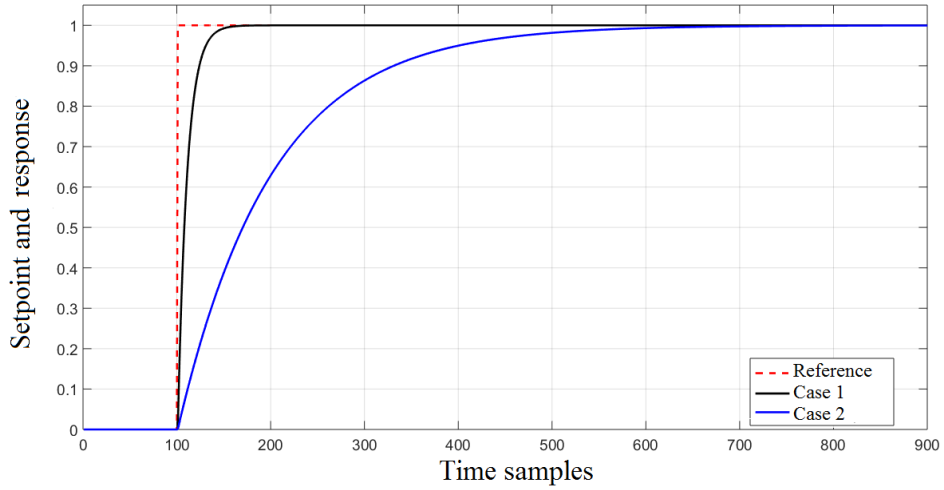

FIGURE 3.10: The aggression of the predictive controller

Case1 on the response graph, shows more aggression compared to case2 which is slower. To achieve this, a constant α is placed in the expression of y^{ref} as follows:

$$y^{ref}(t+k) = \alpha y^{ref}(t+k-1) + (1-\alpha) y^{sp}(t+k) \tag{3.55}$$

α is a value between zero and one and technically (from the expression of y^{ref}), should result in aggressive response when closer to zero and sluggish response when closer to one.

3.2.3 Output prediction [82, 83]

For this research, the Dynamic Matrix Control algorithm (DMC) is used. In making predictions, DMC uses a step response model and typically, a discrete transfer function. DMC as stated earlier, has gained popularity as a control strategy in industry, especially the petrochemical industry. The following subsections, which study derivations of DMC are referenced to [82, 83] and are based on the fact that if considering a unitary finite step input $\vec{u}$, the system output can be represented by equation (3.56)

$$\vec{y} = [0, s(1), s(2), s(3), \ldots, s(n), s(n) \ldots]^T \tag{3.56}$$

where s is the step response output. It is assumed that $s(0) = 0$ and also that the steady state is maintained after transient i.e. $s(n+1) = s(n+2) = s(\infty)$.

Input $\vec{u}$ where $\vec{u} = [u(0), u(1), u(2), u(3), \ldots]^T$ can be written as an addition of steps, equation (3.57)

$$\begin{aligned} \vec{u} =& [1,1,1,1,\ldots]^T \times u(0) + [0,1,1,1,\ldots]^T \times (u(1) - u(0)) \\ &+ [0,0,1,1,\ldots]^T \times (u(2) - u(1)) + \ldots \end{aligned} \tag{3.57}$$

where $\vec{u}$ remains a vector of repeating $u(0)$ for a step input. If $\Delta u(k) = u(k) - u(k-1)$, then the output can be written as

$$\begin{aligned} \vec{y} &= [y(0), y(1), y(2), y(3), \ldots]^T \\ &= [0, s(1), s(2), s(3), \ldots, s(n), s(n), s(n), \ldots]^T \times u(0) \\ &\quad + [0, 0, s(1), s(2), \ldots, s(n-1), s(n), s(n), \ldots]^T \times \Delta u(1) \\ &\quad + [0, 0, 0, s(1), \ldots, s(n-2), s(n-1), s(n), \ldots]^T \times \Delta u(2) \end{aligned} \tag{3.58}$$

It therefore follows that

$$\begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ y(3) \\ \vdots \\ y(n) \\ \vdots \end{bmatrix} = \begin{bmatrix} 0 \\ s(1)u(0) \\ s(2)u(0) + s(1)\Delta u(1) \\ s(3)u(0) + s(2)\Delta u(1) + s(1)\Delta u(2) \\ \vdots \\ s(n)u(0) + s(n-1)\Delta u(1) + \ldots + s(1)\Delta u(n-1) \\ \vdots \end{bmatrix} \tag{3.59}$$

The output at time instant (k) can be represented by equation (3.60)

$$\begin{aligned} y(k) &= \textstyle\sum_{i=1}^{\infty} s(i)\Delta u(k-i) \\ &= s(n)u(k-n) + \textstyle\sum_{i=1}^{n-1} s(i)\Delta u(k-i) \end{aligned} \tag{3.60}$$

Output predictions can be seen as having two parts, i.e. a free response part and a forced response part such that predicted output is the sum of the free response prediction and the forced response prediction. Free response is the system response when the input is unchanged i.e. $\Delta u(k) = 0$ and the forced response is the system response caused by changing input. Multiple step ahead predictions can be presented by writing the individual predictions as in equation (3.61), which considers past input:

$$\begin{aligned}
y(k|k) &= \textstyle\sum_{i=1}^{n-1}(s(i)\Delta u(k-i)) + s(n)u(k-n) \\
y(k+1|k) &= \textstyle\sum_{i=1}^{n-1}(s(i)\Delta u(k+1-i)) + s(n)u(k+1-n) \\
&= \textstyle\sum_{i=2}^{n-1}(s(i)\Delta u(k+1-i)) + s(n)u(k+1-n) + s(1)\Delta u(k|k) \\
y(k+2|k) &= \textstyle\sum_{i=1}^{n-1}(s(i)\Delta u(k+2-i)) + s(n)u(k+2-n) \\
&= \textstyle\sum_{i=3}^{n-1}(s(i)\Delta u(k+1-i)) + s(n)u(k+2-n) + s(1)\Delta u(k-1|k) \\
&\quad +s(2)\Delta u(k|k) \\
&\vdots \\
y(k+n-1|k) &= \textstyle\sum_{i=1}^{n-1}(s(i)\Delta u(k+n-1-i)) + s(n)u(k-1) \\
&= s(n)u(k-1) + \textstyle\sum_{i=1}^{n-1}(s(i)\Delta u(k-i+n-1|k)) \\
y(k+n|k) &= \textstyle\sum_{i=1}^{n-1}(s(i)\Delta u(k+n-i)) + s(n)u(k) \\
&= s(n)u(k-n) + \textstyle\sum_{i=1}^{n}(s(i)\Delta u(k-i+n|k))
\end{aligned} \tag{3.61}$$

The forced response (a function of the current or future changes in the input variable) can be noted separately from the free response (which is a function of the past inputs) can be observed and denoted as f according to

$$f(k|k) = \sum_{i=1}^{n-1}(s(i)\Delta u(k-i)) + s(n)u(k-n) \tag{3.62}$$

$$f(k+1|k) = \sum_{i=2}^{n-1}(s(i)\Delta u(k+1-i)) + s(n)u(k+1-n) \tag{3.63}$$

$$f(k+2|k) = \sum_{i=3}^{n-1}(s(i)\Delta u(k+1-i)) + s(n)u(k+2-n) \tag{3.64}$$

$$f(k+n-1|k) = s(n)u(k-1) \tag{3.65}$$

Vector $\vec{f}$ (3.66) can be used to represent the predictions of the free response at time instant k

$$\vec{f}(k) = [f(k|k), f(k+1|k), f(k+2|k), \ldots, f(k+n-1|k)]^T \tag{3.66}$$

While input is unchanged, the free response equals the system response ($f(k+1|k) = y(k+i)$). Also, from the last predicted time point, $(k+n-1)$ going ahead, the free

response is constant. The multi-step presentation can be written as

$$\begin{aligned}
&y(k|k) = f(k|k)\\
&y(k+1|k) = f(k+1|k) + s(1)\Delta u(k|k)\\
&y(k+2|k) = f(k+2|k) + s(1)\Delta u(k+1|k) + s(2)\Delta u(k|k)\\
&y(k+3|k) = f(k+3|k) + s(1)\Delta u(k+2|k) + s(2)\Delta u(k+1|k) + s(3)\Delta u(k|k)\\
&\vdots
\end{aligned} \tag{3.67}$$

In matrix form, equation (3.68) can represent the expression above

$$\begin{bmatrix} y(k+1|k) \\ y(k+2|k) \\ \vdots \\ y(k+n|k) \end{bmatrix} = \begin{bmatrix} f(k+1|k) \\ f(k+2|k) \\ \vdots \\ f(k+n|k) \end{bmatrix} + \begin{bmatrix} s(1) & 0 & \cdots & 0 \\ s(2) & s(1) & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ s(n) & s(n-1) & \cdots & s(1) \end{bmatrix} \cdot \begin{bmatrix} \Delta u(k|k) \\ \Delta u(k+1|k) \\ \vdots \\ \Delta u(k+n-1|k) \end{bmatrix} \tag{3.68}$$

where $\vec{f}(k+1|k)$ is the free response prior to knowledge of $\Delta u(k|k)$. Equation (3.68) can be written in short as (3.69)

$$\hat{y}(k+1) = \mathbf{M}\vec{f}(k) + \mathbf{G}\Delta\mathbf{u}(k) \tag{3.69}$$

where $\mathbf{M}\vec{f}$ is the free response and $\mathbf{G}$, which is part of the forced response expression, is the dynamic matrix and determines how current and future changes in input affect the output. $\mathbf{G}$ is made up of p rows and c columns, of the step response model (closed-loop stabilised model as required by DMC) values of the system. The closed-loop stable step response for the vertical control system, obtained from system identification, is given by the data points shown partly in Figure 3.11.

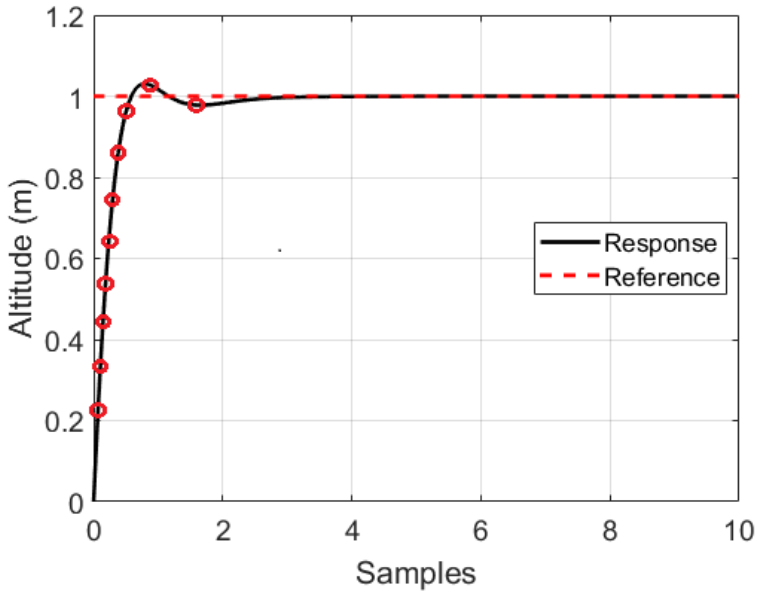

FIGURE 3.11: Step response model for DMC

When 40 sample points are used, this model can be written as:

$$y(t) = \sum_{i=1}^{40} g_i \Delta u(t-i) \tag{3.70}$$

Now, the resulting dynamic matrix can be constructed from:

$$\mathbf{G} = \begin{bmatrix} s_1 & 0 & \cdots & 0 \\ s_2 & s_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ s_c & s_{c-1} & \cdots & s_1 \\ \vdots & \vdots & \ddots & \vdots \\ s_p & s_{p-1} & \cdots & s_{p-c+1} \end{bmatrix} \tag{3.71}$$

To obtain the actual system output in simulations, a discrete transfer function of the system is typically used. Based on this transfer function, the idea is to express the current prediction in terms of the past output and current input. The standard method therefore, as found in [82, 84] and considering the available system follows this procedure:

$$G(z) = \frac{0.01}{z^{-2} - 1.80z^{-1} + 0.86} \tag{3.72}$$

is written in the form:

$$\hat{y}_i = \sum_{j=1}^{d} a_j \hat{y}_{i-1} + \sum_{j=1}^{n} b_j u_i \tag{3.73}$$

where d and n are the number of available terms, a_j and b_j are the coefficients of the denominator and the numerator of the transfer function, such that the actual output is expressed as:

$$\begin{aligned} \hat{y}_i =& | -1.80 + 0.86 | \hat{y}_{i-1} + 0.01 u_i \\ =& 0.94 \hat{y}_{i-1} + 0.01 u_i \end{aligned} \tag{3.74}$$

This method slightly compromises accuracy hence reliability thus, in this thesis, a slight modification is applied to obtain actual output from the full model, expressed as:

$$\hat{y}_i = -1.8\hat{y}_{i-1} + 0.86\hat{y}_{i-2} + 0.01u_i \tag{3.75}$$

3.2.4 Free response recursion

At time sample k, $\vec{f}$ is given by equation (3.76) [82]

$$\vec{f}(k) = [f(k|k), f(k+1|k), f(k+2|k), \cdots, f(k+n-1)|k)]^T \tag{3.76}$$

At instant $k+1$, the vector of f is given by equation (3.77)

$$\vec{f}(k+1) = [f(k+1|k), f(k+2|k), \cdots, f(k+n-1)|k), f(k+n-1)|k)]^T + s\Delta u(k) \quad (3.77)$$

The term $f(k+n-1)|k)$ is repeated because the system transient is assumed to reach steady state at the end of n steps. The term $s\Delta u(k)$ is the change caused by the step input Δu applied at k. Equation (3.77) can therefore be written as

$$\vec{f}(k+1) = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ & 0 & 1 & \ddots & \vdots \\ \vdots & & 0 & \ddots & 0 \\ & & & \ddots & 1 \\ 0 & & \cdots & & 1 \end{bmatrix} \cdot \vec{f}(k) + \begin{bmatrix} s(1) \\ s(2) \\ s(3) \\ \vdots \\ s(n) \end{bmatrix} \cdot \Delta u(k) \quad (3.78)$$

The equation above can be written as equation (3.79)

$$\vec{f}(k+1) = \mathbf{M}\vec{f}(k) + \vec{s}\Delta u(k) \quad (3.79)$$

3.2.5 Adding disturbance

For practical purposes, it is necessary to consider disturbance in the model. The control problem arising is such as that shown in Figure 3.12. This introduces measured disturbance d and unmeasured disturbance w (which, includes model inaccuracy).

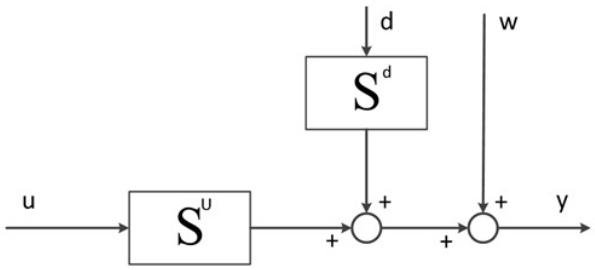

FIGURE 3.12: Scheme of the DMC algorithm including disturbance

where S^u and S^w model the effects of u on y and d on y respectively. For the initial computations of the free response $\vec{f}$, it is assumed that disturbance and control step $\Delta u(k)$ are equal to zero. The output is a summation of:

- predicted free response ($\vec{f}(k+1)$), as in (3.68)
- forced response ($\mathbf{S}^u\Delta\mathbf{u}$)
- forced response ($\mathbf{S}^d\Delta\mathbf{d}$)
- the response due to w

Considering the prediction horizon p (p = number of predictions) and available information to the point k, the resultant equation is (3.80) for p-step ahead prediction [82, 83]

$$\hat{y} = \vec{f}(k+1) + \mathbf{S}^u\vec{\Delta u} + \mathbf{S}^d\vec{\Delta d} + \vec{\omega} \tag{3.80}$$

This is expanded to (3.81)

$$\begin{bmatrix} \hat{y}(k+1|k) \\ \hat{y}(k+2|k) \\ \vdots \\ \hat{y}(k+p|k) \end{bmatrix} = \begin{bmatrix} f(k+1|k) \\ f(k+2|k) \\ \vdots \\ f(k+p|k) \end{bmatrix} + \begin{bmatrix} s^u(1) & 0 & \cdots & 0 \\ s^u(2) & s^u(1) & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ s^u(p) & s^u(p-1) & \cdots & s^u(1) \end{bmatrix} \cdot \begin{bmatrix} \Delta u(k|k) \\ \Delta u(k+1|k) \\ \vdots \\ \Delta u(k+p-1|k) \end{bmatrix}$$
$$+ \begin{bmatrix} s^d(1) & 0 & \cdots & 0 \\ s^d(2) & s^d(1) & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ s^d(p) & s^d(p-1) & \cdots & s^d(1) \end{bmatrix} \cdot \begin{bmatrix} \Delta d(k|k) \\ \Delta d(k+1|k) \\ \vdots \\ \Delta d(k+p-1|k) \end{bmatrix}$$
$$+ \begin{bmatrix} w(k+1|k) \\ w(k+2|k) \\ \vdots \\ w(k+p|k) \end{bmatrix} \tag{3.81}$$

In (3.80), the matrices $\mathbf{S}^u$ and $\mathbf{S}^d$ have p rows and if $n < p$ the missing elements in the step responses $\mathbf{S}^u$ and $\mathbf{S}^d$ are then duplicates of the last values $S^u(n)$ and $S^d(n)$ of the corresponding step response model, as it was assumed earlier that the transient ends after n instants. To calculate the change in measured disturbance at instant k, the relationship ($\Delta d(k|k) = d(k) - d(k-1)$) is used. A few assumptions need to be made now to cover for Δd and ω which are unknown. To this end, it assumed that:

- Measured disturbance is constant ($\Delta d(k+1) = \Delta d(k+2) = ... = 0$)
- Unmeasured disturbance is the difference between the model predicted output and the actual output ($\omega(k|k) = y(k) - \hat{y}(k)$). The predicted output can be represented as $f(k|k)$ and $\hat{y}(k) = f(k|k)$.
- Unmeasured output also remains unchanged in future, ($\omega(k+1|k) = \omega(k+2|k) = ... = \omega(k+p|k)$).

It is further assumed that the measurements in output and in disturbance are free of noise, although in simulation, noises will be taken into account. Equation (3.81) can therefore be written as equation (3.82)

$$\begin{bmatrix} \hat{y}(k+1|k) \\ \hat{y}(k+2|k) \\ \vdots \\ \hat{y}(k+p|k) \end{bmatrix} = \begin{bmatrix} f(k+1|k) \\ f(k+2|k) \\ \vdots \\ f(k+p|k) \end{bmatrix} + \begin{bmatrix} s^u(1) & 0 & \cdots & 0 \\ s^u(2) & s^u(1) & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ s^u(p) & s^u(p-1) & \cdots & s^u(1) \end{bmatrix} \cdot \begin{bmatrix} \Delta u(k|k) \\ \Delta u(k+1|k) \\ \vdots \\ \Delta u(k+p-1|k) \end{bmatrix} + \begin{bmatrix} s^d(1) \\ s^d(2) \\ \vdots \\ s^d(p) \end{bmatrix} \cdot \Delta d(k|k) + \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \cdot (y(k) - f(k|k)) \end{bmatrix} \tag{3.82}$$

Equation (3.82) can be arranged as equation (3.83) where the prediction is a sum of the past, present and the future [83]

$$\underbrace{\hat{y}(k+1)}_{\text{=prediction}} = \underbrace{\mathbf{T}\vec{f}(k)}_{\text{=past}} + \underbrace{\mathbf{s^d}\Delta d(k) + (y(k) - f(k|k))}_{\text{=present}} + \underbrace{\mathbf{G}\vec{\Delta u}(k)}_{\text{=future}} \tag{3.83}$$

The idea then, is that the prediction comprises some free response $\mathbf{T}\vec{f}$ due to the behavior of the system in the past, some measured disturbance (which is fed forward) and some measurement bias (which is fed back) both based on the current state of the system, and then finally, some component due to the future control action to be applied to the system after the optimisation problem is solved. The matrix $\mathbf{s^d}$ is obtained by performing a step test to model the effect of disturbance on the system then, in the same way as for the dynamic matrix, coefficients of $\mathbf{s^d}$ are obtained, with the same dimensions.

3.2.6 Free response

To this point, the free response $\vec{f}$ has been a column vector as equation (3.66). Equation (3.83) introduced the $p \times n$ matrix $\mathbf{T}$ which is characterised as follows [82]:

- when $p > n$, $\mathbf{T}$ displaces vector f and repeats the last element
- when $p = n$, $\mathbf{T}$ displaces vector f and repeats the last element once
- when $p < n$, $\mathbf{T}$ displaces vector f and cuts it to have p elements

$$\mathbf{T}_{p\times n} = \begin{bmatrix} 0 & 1 & 0 & & \cdots & 0 \\ 0 & 0 & 1 & \ddots & & 0 \\ \vdots & & 0 & \ddots & 0 & \vdots \\ 0 & & & \ddots & 1 & 0 \\ 0 & & \cdots & & 0 & 1 \\ \vdots & & & & \vdots & \vdots \\ 0 & & \cdots & & 0 & 1 \end{bmatrix}_{p>n} \tag{3.84}$$

$$\mathbf{T}_{p\times n} = \begin{bmatrix} 0 & 1 & 0 & & \cdots & 0 \\ 0 & 0 & 1 & \ddots & & 0 \\ \vdots & & 0 & \ddots & 0 & \vdots \\ 0 & & & \ddots & 1 & 0 \\ 0 & & \cdots & & 0 & 1 \\ 0 & & \cdots & & 0 & 1 \end{bmatrix}_{p=n} \tag{3.85}$$

$$\mathbf{T}_{p\times n} = \begin{bmatrix} 0 & 1 & 0 & & \cdots & & 0 & \cdots & 0 \\ 0 & 0 & 1 & \ddots & & & & & 0 \\ \vdots & & 0 & \ddots & 0 & & & & \vdots \\ 0 & & & \ddots & 1 & 0 & \cdots & & 0 \\ 0 & & \cdots & & 0 & 1 & 0 & \cdots & 0 \end{bmatrix}_{p<n} \tag{3.86}$$

To consider a case with a manipulated variable and some measured disturbance, it is necessary to update the expression (3.79) to include a feedfoward term for the free response(3.87) [82]

$$\mathbf{f}(k+1)) = \underbrace{\begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ & 0 & 1 & \ddots & & \vdots \\ \vdots & & 0 & \ddots & & \\ & & & \ddots & 1 & 0 \\ 0 & & \cdots & & 0 & 1 \\ 0 & & \cdots & & 0 & 1 \end{bmatrix}}_{\mathbf{M}} \vec{f}(k) + \underbrace{\begin{bmatrix} s^u(1) \\ s^u(2) \\ s^u(3) \\ \vdots \\ s^u(n) \end{bmatrix}}_{\text{su}} \Delta u(k) + \underbrace{\begin{bmatrix} s^d(1) \\ s^d(2) \\ s^d(3) \\ \vdots \\ s^d(n) \end{bmatrix}}_{\text{sd}} \Delta d(k) \tag{3.87}$$

(3.87) can be represented as (3.88)

$$\mathbf{f}(k+1) = \mathbf{M}\vec{f}(k) + \vec{s^u}\Delta u(k) + \vec{s^d}\Delta d(k) \tag{3.88}$$

where $\mathbf{M}$ is an $n \times n$ matrix, and $\vec{s^u}$ and $\vec{s^d}$ have the same number of step coefficients.

3.2.7 Control horizon

The final element to consider before looking into the realisation of the cost function, is the control horizon c. Introducing c means that only c future control moves are considered in the optimization while the remaining changes will be zeros [82, 83]

$$\Delta u(k+c|k) = \Delta u(k+c+1|k) = \cdots = \Delta u(k+c+p-1|k) = 0 \tag{3.89}$$

Prediction of output can therefore be written as (3.90)

$$\begin{bmatrix} \hat{y}(k+1|k) \\ \hat{y}(k+2|k) \\ \vdots \\ \hat{y}(k+p|k) \end{bmatrix} = \begin{bmatrix} f(k+1|k) \\ f(k+2|k) \\ \vdots \\ f(k+p|k) \end{bmatrix} + \underbrace{\begin{bmatrix} s^u(1) & 0 & \cdots & 0 \\ s^u(2) & s^u(1) & \ddots & \vdots \\ & s^u(2) & \ddots & 0 \\ \vdots & & \ddots & s^u(1) \\ & \vdots & & s^u(2) \\ & & & \vdots) \\ s^u(p) & s^u(p-1) & \cdots & s^u(p-c+1) \end{bmatrix}}_{\mathbf{G}} \cdot \begin{bmatrix} \Delta u(k|k) \\ \Delta u(k+1|k) \\ \vdots \\ \Delta u(k+c-1|k) \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \begin{bmatrix} s^d(1) \\ s^d(2) \\ \vdots \\ s^d(p) \end{bmatrix} \cdot \Delta d(k|k) + \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1\cdot \\ 1 \end{bmatrix} (y(k) - f(k|k)) \tag{3.90}$$

This can be summarised by (3.91)

$$\hat{y}(k+1) = \underbrace{\mathbf{T}\vec{f}(k) + \vec{s^d}\Delta d(k) + (y(k) - f(k|k))}_{\text{yp(k+1)}} + \mathbf{G}\vec{\Delta u}(k) \tag{3.91}$$

Equation (3.91) simplifies to (3.92)

$$\hat{y}(k+1) = \vec{y^p}(k+1) + \mathbf{G}\vec{\Delta u}(k) \tag{3.92}$$

The term y^p where ($y^p(k+1) = [y^p(k+1), y^p(k+2), \cdots, y^p(k+p)]^T$) represents the past free response term and the present feedforward and feedback terms. $\mathbf{G}\vec{\Delta u}$ where $\mathbf{G}$ is of $p \times c$ dimension is due to the future control moves.

3.2.8 Optimisation

The DMC algorithm is intended to determine the sequence of control action $\vec{\Delta u}$ that will cause a forced response $\mathbf{G}\vec{\Delta u}$ in a manner such that the error between the output of the system and the desired reference is minimum. It is common to define the future reference sequence by a set point y^{sp}. The task is therefore to find

$$\mathbf{G}\vec{\Delta u} + \vec{y^p} = \vec{y^{sp}} \tag{3.93}$$

This problem has p equations and c unknowns and when $p = c$, a unique solution exists to the equation, while, when c is less than p, which would likely be the case, there is generally no exact solution. In this case, it helps to find a vector $\vec{\Delta u}$ which optimally minimises the sum of squared errors $\vec{\epsilon}$

$$\vec{\epsilon} = \mathbf{G}\vec{\Delta u} - (\vec{y^{sp}} - \vec{y^{p}}) \tag{3.94}$$

The cost function to be minimised is written as [82]

$$J = \vec{\epsilon}^T \vec{\epsilon} \tag{3.95}$$

This quadratic cost function can then be solved by setting the derivative to zero, which would correspond to the minimum [82].

$$\frac{\partial J}{\partial \vec{\Delta u}} = 2\mathbf{G}^T[\mathbf{G}\vec{\Delta u} - (\vec{y^{sp}} - \vec{y^{p}})] = 0 \tag{3.96}$$

$$\mathbf{G}^T\mathbf{G}\vec{\Delta u} = \mathbf{G}^T(\vec{y^{sp}} - \vec{y^{p}}) \tag{3.97}$$

$$\vec{\Delta u} = [\mathbf{G}^T\mathbf{G}]^{-1}\mathbf{G}^T(\vec{y^{sp}} - \vec{y^{p}}) \tag{3.98}$$

Equation (3.98) represents a solution, which is the minimum if the second derivative $\mathbf{G}^T\mathbf{G}$ is 'positive definite', where some matrix $\mathbf{M}$ is said to be positive definite if $x^T\mathbf{A}x \geq 0$ for all $x \neq 0$. The control moves of the predictive controller can be weighed by adding a factor r:

$$J = \vec{\epsilon}^T\vec{\epsilon} + \vec{\Delta u}^T r\vec{\Delta u} \tag{3.99}$$

The solution, which represents the DMC formulation, is then written as equation (3.100)

$$\begin{aligned} \vec{\Delta u} &= \underbrace{[\mathbf{G}^T\mathbf{G} + r\mathbf{I}]^{-1}\mathbf{G}^T}_{\mathbf{H}}\underbrace{(\vec{y^{sp}} - \vec{y^{p}})}_{\mathbf{e}} \\ &= \mathbf{H}\vec{e} \end{aligned} \tag{3.100}$$

With a value for r, $[\mathbf{G}^T\mathbf{G} + r\mathbf{I}]^{-1}\mathbf{G}^T$ can be calculated for the unconstrained problem. The constrained problem is an extension to the unconstrained case as the necessary derivations are spelt out in texts such as [82] and [83].

3.2.9 DMC implementation

The design process involves selection of the controller sampling time and horizons (prediction and control horizon), specifying the desired constraints, scaling factors and weights.

Prediction Horizon

The prediction horizon, p, is the number of control intervals ahead from the current time interval k, which the controller must evaluate when computing control action at interval k. The prediction horizon p is selected and held constant early in the controller design while tuning other controller settings. The value of p is selected according to the rule-of-thumb that given the desired closed-loop response time (T) $T \simeq pT_s$ [96].

Control horizon

The control horizon c, is the number of moves in the control action to be computed by the controller at the time k. The control horizon falls between 1 and the prediction horizon p. Keeping c small means fewer variables need to be calculated by the Quadratic Program (QP) at each control interval [96]. This in turn, allows faster computation.

Sampling time

It is recommended to set the sampling time T_s initially and the hold it constant while other controller parameters are adjusted and typically, T_s is set between 10 percent and 20 percent of the minimum desired closed-loop response time [82] although this applies mainly to process control. Much lower T_s is used for the quadrotor system given the faster dynamics. Technically, as T_s decreases, unmeasured disturbance rejection improves, however, the computational demand increases too. The task is therefore to balance performance and the computational expense.

3.2.10 Tuning the model predictive controller

Typically, tuning the model predictive controller is not standardised but rather more intuitive. Certain aspects have however come to realisation, aspects that provide some guideline as to approximation of controller parameters. In order to get reasonable reference points for the tuning parameters, these typical guidelines will be followed. To do this, the system model will be converted (approximated) into a First Order Plus Time Delay (FOPTD) model, which is represented by the differential equation:

$$\tau_p \frac{dy}{dt} + y(t) = K_p u(t - t_d) \tag{3.101}$$

where τ_p is a time constant, K_p is the process gain and t_d is delay time. The time constant is the amount of time taken for the output to reach 63.2 percent of the steady-state condition. The process gain is the change in the output, caused by a unit change in the input. The time delay can be defined as the time shift in the input variable. The FOPTD models are very useful because they approximate systems of higher order with acceptable deviation, in most cases. The transfer function for the differential equation is:

$$G(s) = \frac{Y(s)}{U(s)} = \frac{K_p e^{-t_d s}}{\tau_p s + 1} \tag{3.102}$$

From the FOPTD model, it is possible to select tuning parameters as follows [96]:

- For the sampling time interval T, time constant τ_p and delay time t_d (defined aboved):

  $$T \leq 0.5 t_d \tag{3.103}$$

  and

  $$T \leq 0.1 \tau_p \tag{3.104}$$

- Discrete dead-time k

  $$k = \frac{t_d}{T} + 1 \tag{3.105}$$

- Prediction horizon p

  $$p = \frac{5\tau_p}{T} + k \tag{3.106}$$

- Control horizon c is selected usually up to a maximum of the value 6.

The FOPTD model for the altitude system is:

$$Z(s) = \frac{1.002 e^{-0.0484 s}}{0.14436 s + 1} \tag{3.107}$$

With these values, the tuning parameters are calculated as shown in Table 3.1

Parameter	**Value**
p	78
c	5

TABLE 3.1: Tuning parameters

To verify that the parameters are in good range, it is possible to use the system step response and some standard criterion [96] as follows. FOPTD models can be derived from a step response according to Figure 3.13:

The criterion states that, from the step response [96]:

$$\tau_p = \frac{1}{0.7}(t_2 - t_1) \tag{3.108}$$

$$t_d = t_1 - 0.4 \tau_p \tag{3.109}$$

The step responses of the initial systems are given in (Appendix C) and produce:

$$\tau_p = \frac{1}{0.7}(0.22 - 0.12) = 0.142 \tag{3.110}$$

$$t_d = 0.12 - 0.4 * 0.142 = 0.063 \tag{3.111}$$

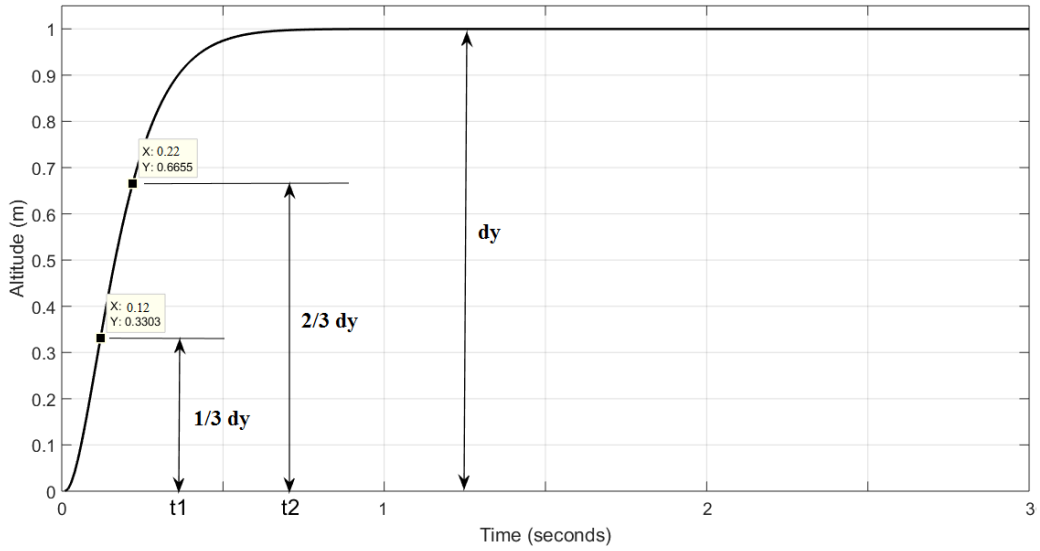

FIGURE 3.13: Derivation of FOPTD from step response

These results are clearly comparable to the values obtained earlier. However, because the step response model used has 70 data points, p is set initially to 65. The parameters used for the initial tuning of the predictive controller are as shown in Table 3.2

Parameter	Value
p	65
c	5

TABLE 3.2: Tuning parameters (initial)

To implement the DMC algorithm efficiently, a flow chart is drawn up and is as shown in Figure 3.14:

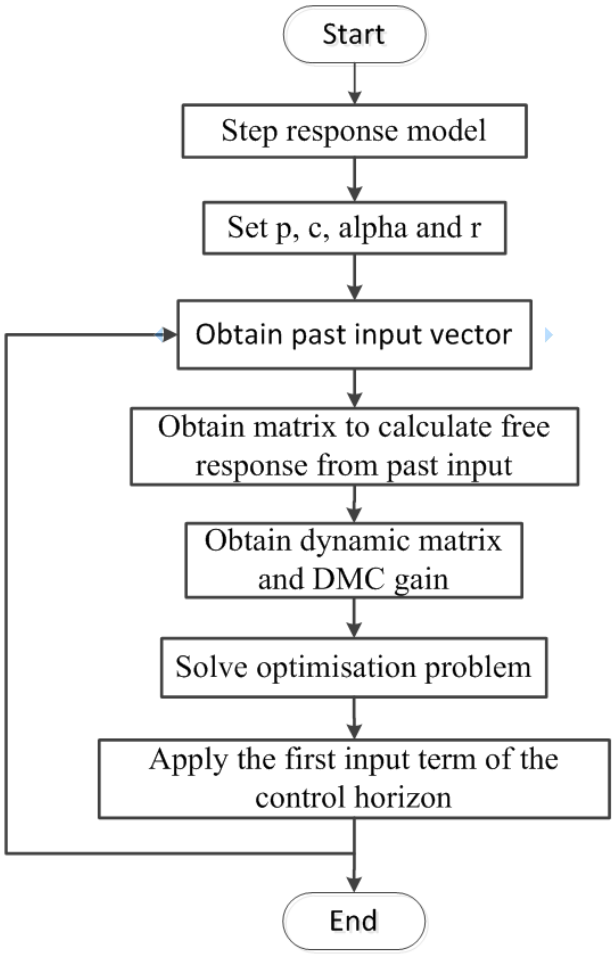

FIGURE 3.14: DMC flow chart

The DMC algorithm can then be implemented. The overall strategy is as follows:

- Optimise p and c
- Tune r and α
- Apply more complex reference signals and investigate the behaviour of the system in each case.

DMC simulation

The initial simulation is run with the set parameters p = 65 and c = 5 while r and α are kept in the middle, i.e. 0.5. The behaviour of the system is characterised by Figure 3.15.

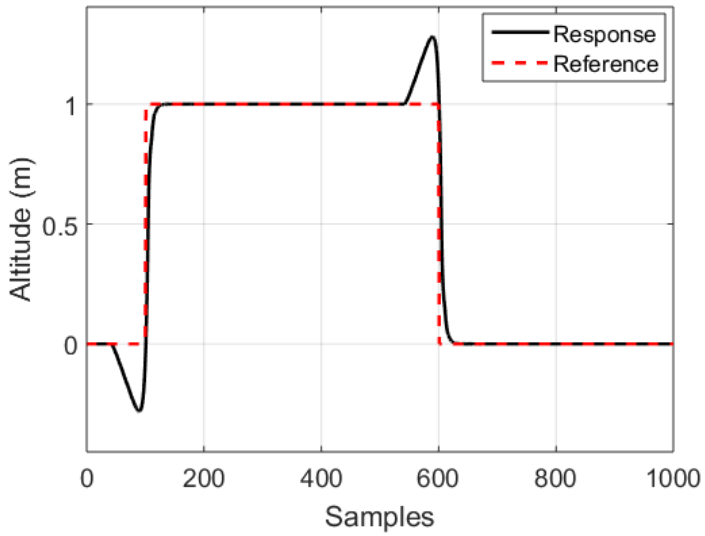

FIGURE 3.15: DMC p=65, c=5, r=0.5 and α=0.5

While the response shows poor performance before change in input, it is clear that the starting point for the tuning of the controller is feasible in reaching the end goal. This poor initial performance is characterised by an undershoot, in altitude, prior to the step input response. This undershoot relates to the non minimum phase behaviour of the quadrotor and must be fixed by adjusting p and c settings. It becomes necessary at this point to discuss the non minimum phase quadrotor behaviour.

Non minimum phase behaviour

The response of a non minimum phase system to some step input is typically characterised by an "undershoot". This means that, given zero initial output and positive step input, the output becomes negative before switching direction and converging to the set positive value. The undershoot most commonly occurs when the control system has an odd number of real right-half plane zeros. In some cases though, non minimum phase could result from time delay in the control system, which is highly

likely in this case. Adjusting the tuning parameters p and c should therefore affect this non minimum phase behaviour, as these parameters affect computation.

DMC simulation

Moving on with DMC simulations, the next simulation is run with $p = 65$ and $c = 10$, and then several other settings until performance is satisfactory. The results are shown by Figure 3.16

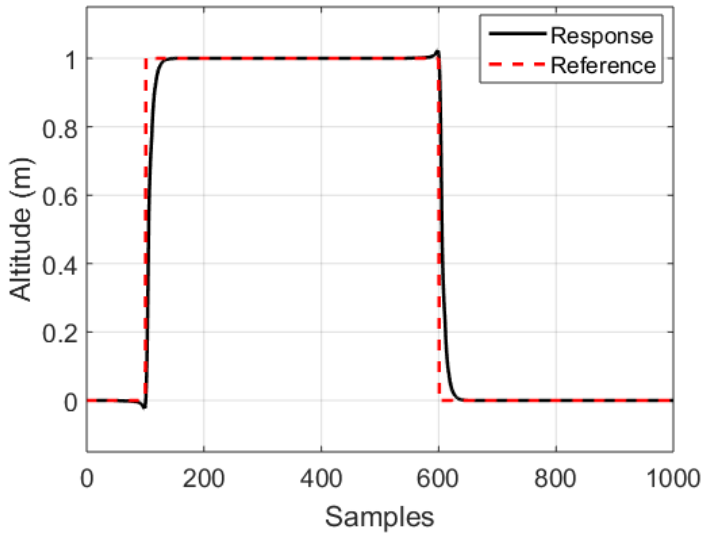

FIGURE 3.16: DMC p=65, c=10, r=0.5 and α=0.5

Increasing the length of the control horizon proves to improve the initial system performance. However, higher values of p and c increase computational expense. For efficiency, the computation expense must kept to a minimum. The preferred value for c is in fact 5 or less and for p, the smallest possible value is desired. A way must be found, to achieve high quality performance with small values of p and c.

In [85], a solid conclusion is reached stating that, in reality, a low and intermediate value of p results in a better performance of a Dynamic Matrix Controller. Based on this, low values of p are tested with $c = 5$. Some of the results are shown by Figure 3.17 to Figure 3.18.

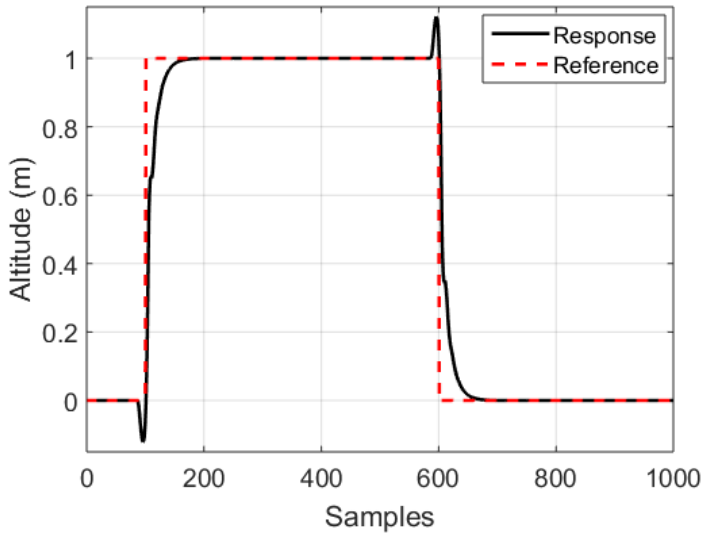

FIGURE 3.17: DMC p=20, c=5, r=0.5 and α=0.5

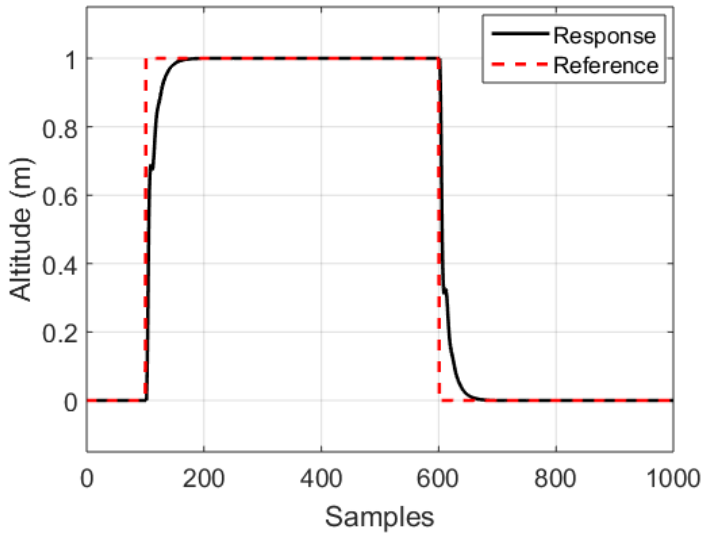

FIGURE 3.18: DMC p=5, c=5, r=0.5 and α=0.5

It is observed that $p = 5$ and $c = 5$ bring about the desired response.

Now, adjusting α must control the aggression (and consequently, the smoothness) of the response. The simulation is therefore run for $p = 5$ and $c = 5$ with $\alpha = 0.2$, $\alpha = 0.7$ and $\alpha = 0.9$. The responses are given by Figure 3.19

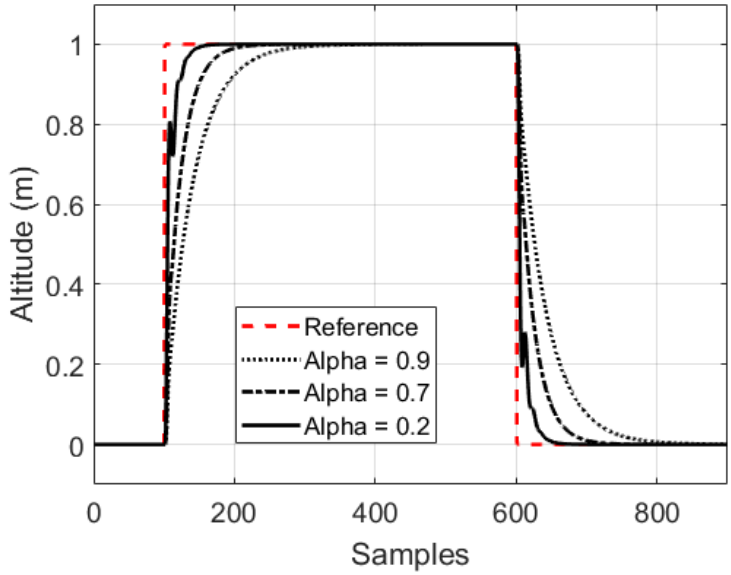

FIGURE 3.19: DMC aggression

It is observed $\alpha = 0.7$ settles to steady-state within 100 samples and gives satisfactory response. $\alpha = 0.7$ is selected as the DMC setting.

3.2.11 More complex reference trajectory

The selected setting based on the simulations carried out so far is $p = 5$, $c = 5$, $r = 0.5$ and $\alpha = 0.7$. This setting, with $\alpha = 0.7$, is examined for different set point trajectories. The ramp response is as shown in Figure 3.20 and a sinusoidal response is as shown in Figure 3.21.

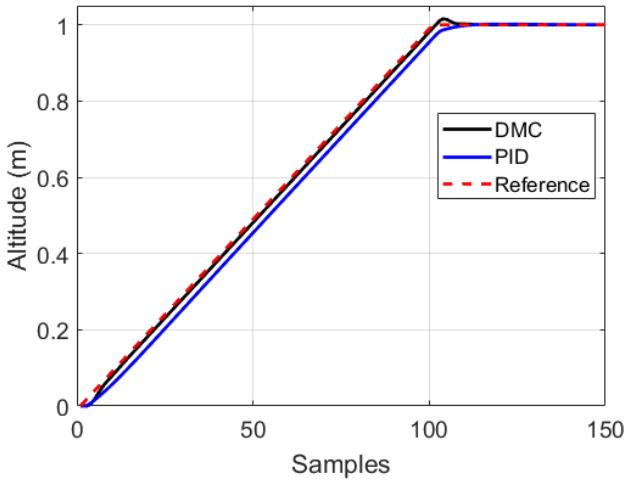

FIGURE 3.20: DMC p=5, c=5, r=0.5 and α=0.7

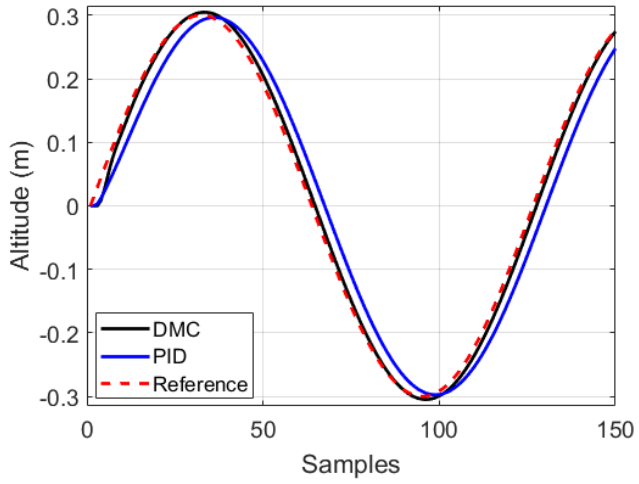

FIGURE 3.21: DMC p=5, c=5, r=0.5 and α=0.7

The selected setting shows higher quality control with DMC. The 3D trajectory tracking with DMC is shown in Figure 3.22.

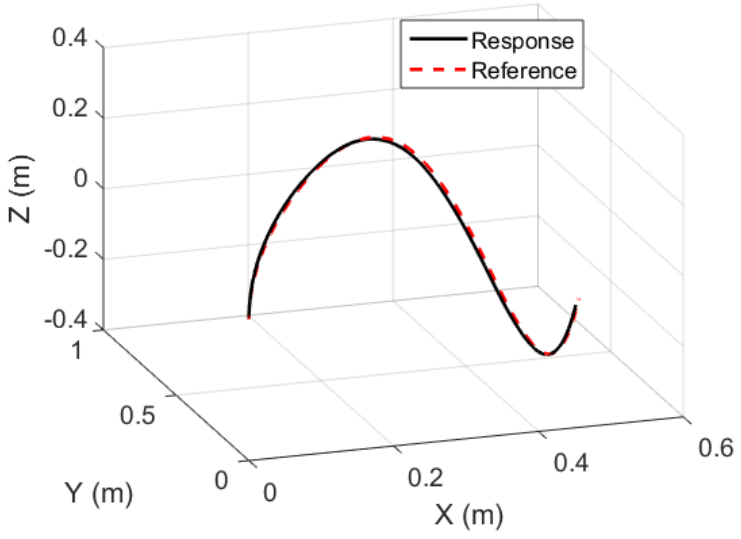

FIGURE 3.22: DMC p=5, c=5, r=0.5 and α=0.7

It can be concluded that DMC, at this point without adaptation (Figure 3.22), achieves superior reference tracking for the quadrotor, especially even as the reference trajectory becomes more complex.

3.2.12 Considering Disturbance

The next step would be to investigate how ADMC handles disturbance. The disturbance, as explained later, considers a load mass being suddenly thrown onto the

support of the quadrotor in hover. Step disturbance (which can be modelled as a change in sensor output [87] or as a force [88]) is used. Disturbance simulations are run and detailed in the next section.

3.3 ADMC implementation

Before discussing the implementation of ADMC, the general structure of the control law must be summarised, from the information already given within this chapter. The first part is a representation of the DMC formulation, which computes the control law:

$$\vec{\Delta u} = [\mathbf{G}^T\mathbf{G} + r\mathbf{I}]^{-1}\mathbf{G}^T(\vec{y^{sp}} - \vec{y^p}) \tag{3.112}$$

This formula is based on an adjustable system model whose parameters are updated by an adaptive law, and given by:

$$\hat{\theta}(t+1) = \hat{\theta}(t) + P(t+1)\varphi(t)\varepsilon(t+1) \tag{3.113}$$

Both equations were derived and explained earlier in this chapter. There are two main ways to implement ADMC. The first form of ADMC is one in which adaptation is used to calculate and update horizon parameters of the DMC controller i.e. the prediction and/or control horizons, depending on the estimated real-time system dynamics. In the work of Klopot, this is achieved by using linear spline interpolation, for a nonlinear hydraulic plant [89]. In a similar way, the suppression and scaling factor can also be adjusted in time [90].

Another (somewhat more complex) realisation of ADMC is one where the actual DMC model is updated. This has been investigated for a distillation column [91] and also for a turbine system [92]. This form of ADMC typically involves utilisation of several models of the system, for different points, then, based on system measurements, finding the best model for a particular operating point, typically using interpolation. This has also been successfully implemented for process control [93, 94, 95]. Other forms of ADMC have been implemented and are in most ways, extensions of the above stated. In this research, the second approach to ADMC is taken because this approach potentially poses higher robustness especially for larger disturbances. Two things, are done differently though. This paper proposes using a single step response model for different disturbances to act on the system, causing change of dynamics. As stated earlier, typically, several models are utilised for different operating points and one is activated at each sampling instant. To do this efficiently, the deactivated models do not compute the control laws but, to ensure bump-less transfer from one model to another, the deactivated controllers are required to continue calculations for the dynamic matrix. This is therefore computationally expensive and even worse for the quadrotor given the fast dynamics and limited computing power. The idea for this research, is therefore to run multiple simulations to attempt to find a single adjustable step response model to use

for different control scenarios resulting from load pick-up, load drop-off or external disturbance (wind), based on analysing how model parameters are affected by disturbance.

Secondly, instead of using interpolation to switch from one model to another, a Supervisor is used. The Supervisor monitors system behaviour and then decides how to adjust the DMC step response model which is used for prediction. The advantages of using a supervisor over interpolation formulae are:

- Code is relatively simple (and supposedly more efficient).
- A RLS estimator already exists, which estimates model parameters in real time, which can then conveniently and readily be used by the supervisor in making decisions.

3.4 Summary

Tuning the ADMC is quite an intuitive process, one which demands extensive simulation. The adaptive controller must be tuned for the available plant (from system identification based on experimental data), then the DMC controller according to the required performance. When tuned well however, the ADMC controller satisfies the two main objectives i.e. adaptation to varying plant dynamics and optimal reference tracking. ADMC simulations can be run from a Matlab file with the necessary mathematical expressions, or from a Simulink model. The latter, while more challenging, does however provide much more flexibility and convenience for model modification and extension. At this point, constraints have not been considered.

Chapter 4

IMPLEMENTATION OF ADMC

ADMC is implemented and evaluated based on some designed experiment to investigate the pick-and-place operation of the delivery quadrotor in simulation. Observations can therefore be made to determine the real performance of ADMC. The idea is to start with simulation tests, then develop the simulation experiment gradually until tests can be conducted experimentally, in a manner that is safe, reliable and able to demonstrate the performance of ADMC (and other controllers) for a load-place task given a delivery quadrotor.

4.1 The Simulation Experiment

Before discussing the implementation of ADMC for the available system, the general experiment to be simulated is described. The experiment involves the quadrotor flying across the room, in picking up a load and dropping the load over some wall as illustrated by Figure 4.1. The load, which initially rests on a holding platform, is hooked by a hook on the bottom of the quadrotor along the central axis, and falls onto the support of the quadrotor until placed. This experiment is simulated from a Matlab file, coded from the RLS and DMC equations described in Chapter 3.

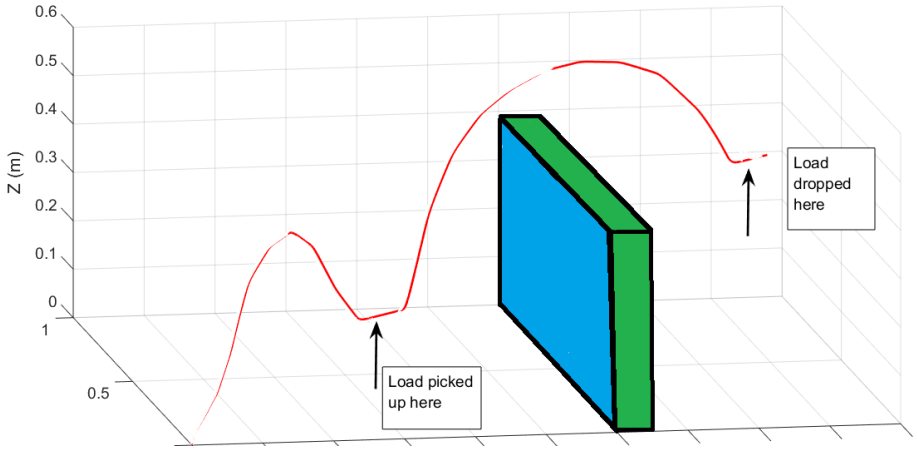

FIGURE 4.1: Pick and place experiment

AMPC in general, uses a fixed model structure, whose parameters are updated with time. Ideally, when the controller makes a prediction, the prediction is based on a model representing the current conditions. The approach taken in implementing AMPC is generally to first design a traditional model predictive controller. This controller is designed based on the model of the quadrotor at nominal operating conditions. The next step is to introduce adaptation to update the system model as the system dynamics change. A supervisor is used to achieve this step. To implement the supervisor, it is necessary to understand how the system dynamics change when the quadrotor, in this case, picks up a load. This is done by running several simulations on the system. The parameters a_1 and a_2 in the subsystems of control change significantly as dynamics change, compared to b_1 and b_2 as a result of load disturbance. Figure 4.2 shows how the parameters change over time, as an example for the altitude subsystem.

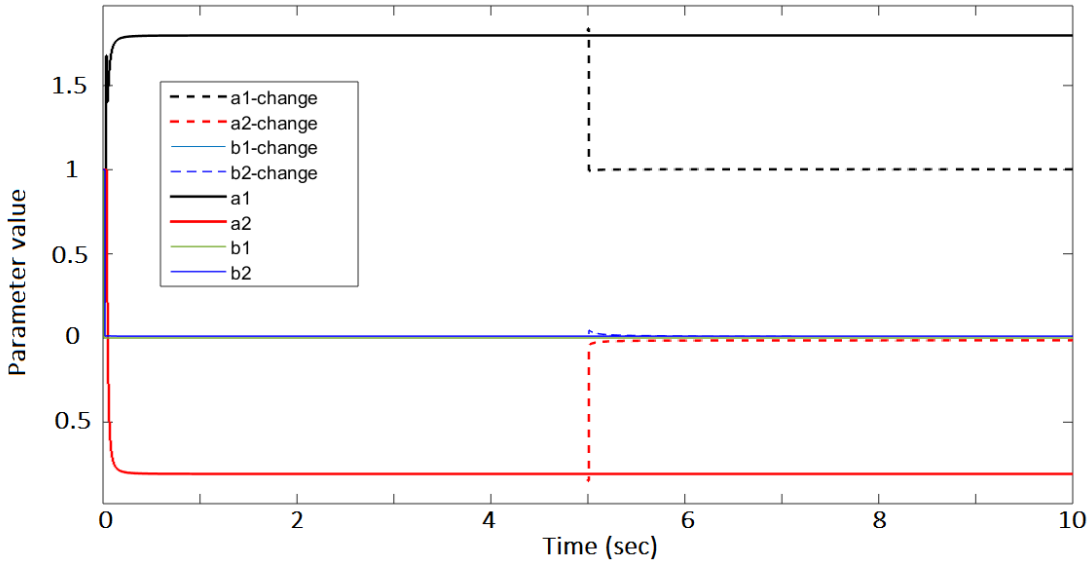

FIGURE 4.2: Parameter change

The plot $a1 - change$ for example shows how the parameter $a1$ changes with load drop-off. To add adaptation to DMC, disturbance step responses are required, which will then be used in computing control action (manipulated variable) to compensate for the disturbance. Different step responses exist for different disturbances as shown by Figure 4.3, for example, which gives responses from load drop-off of two different weights.

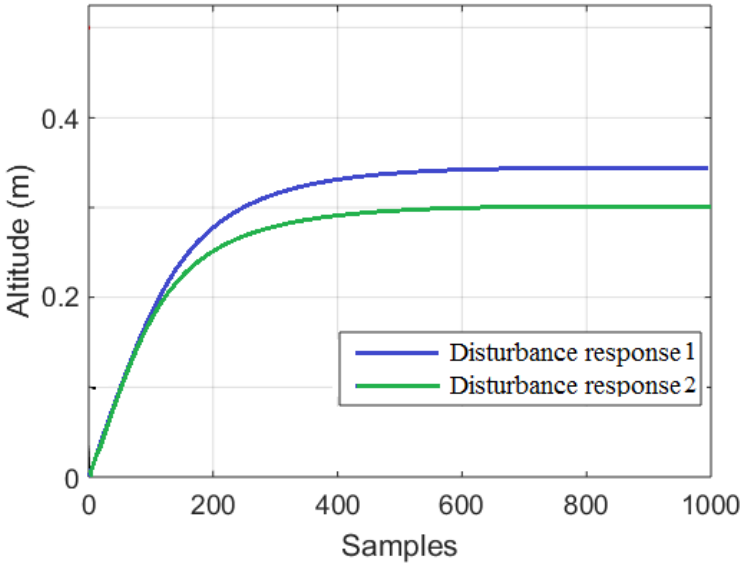

FIGURE 4.3: Disturbance step response

The disturbance step response is considered from hover, with some load pre-attached from take-off.

Adding adaptation to DMC

It is intended to correct for the load pick up using one step response model, and not multiple individual response models. This means that one step response model is used for different load disturbances. This model must however be scaled, hence adjusted according to disturbance magnitude and direction. A scaling factor (k) which is dependent on the magnitude and direction of change caused by the disturbance is proposed. It is necessary to therefore introduce a reliable method to adjust the scaling factor according to the measurements of the RLS estimator. Supervision is proposed as detailed next.

4.1.1 Supervision

Supervisory control is a method of control based on switching. Mathematical logic is applied to improve system response quality especially in cases where a controller designed for one operating point does not meet the required quality [98, 99]. In close relation to supervisory control, is variable structure control (VSC). The VSC method is based on changing the dynamics of a non-linear system through some high-frequency switching process. The simplest illustration of control supervision is that of an industrial system with a human technician who periodically adjusts the set points of some number of PID controllers to best account for changes in the operational environment. The switching is based on decisions made by the supervisor (monitor), from the result of processing some output and input data from the

system. Figure 4.4 shows the general architecture of supervisory control, in this case switching between only two controllers.

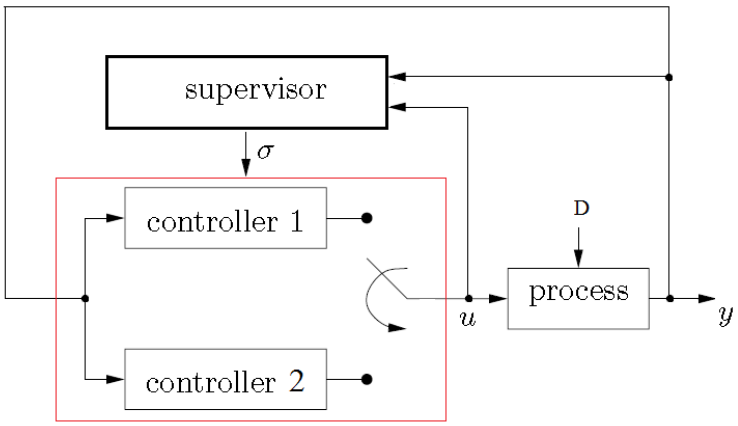

FIGURE 4.4: Architecture of supervisory control

In Figure 4.4, D represents the disturbance signal while σ represents the switching signal. The main types of supervision are as follows [99]:

1. Sequenced supervision - this follows a defined sequence in trying out one controller after the other until acceptable performance in reached.
2. Performance-based supervision - this uses some available data and decides a controller to switch to only when performance becomes unsatisfactory according to some set performances indices or index.
3. Estimator-based supervision - this uses some observed data to estimate the process model and then select a controller based on the estimated model.

In this work, estimator-based supervision is proposed because an estimator is available already. Based on the estimated parametric value, some scaling factor (k) of the step response model is adjusted. The way this is implemented is that the step response model (DMC internal) is prefixed by a scaling factor (k):

$$Response_{step} = k * [0, s(1), s(2), s(3), s(4), ...] \tag{4.1}$$

Now, changing the sign of k allows for the consideration of opposite disturbance effects (quadrotor load drop-off or pick up). Changing the value (magnitude) of k then allows for the consideration of different load sizes (weights). To make this useful, it is then necessary to get an appreciation of how the actual parameters of the estimated model vary according to different loads, and also whether the load is dropped or picked. The description of Figure 4.2 makes this possible. Adjusting k is therefore a matter of mapping the change in the parameters. As stated earlier in this chapter, the parameters a_1 and a_2 change significantly with loading and offloading of the quadrotor. These parameters change almost equally, relying on either is sufficient in determining loading on the quadrotor. The parameter a_1 is used, where,

knowing how the parameter changes when the quadrotor drops or picks a load, is used to adjust k. In implementation, 'if' commands are employed.

The theory of RLS parameter estimation has been covered in the first section of this chapter. The final proposed control scheme is illustrated by Figure 4.5, where the 'k' block represents the scaling factor explained above.

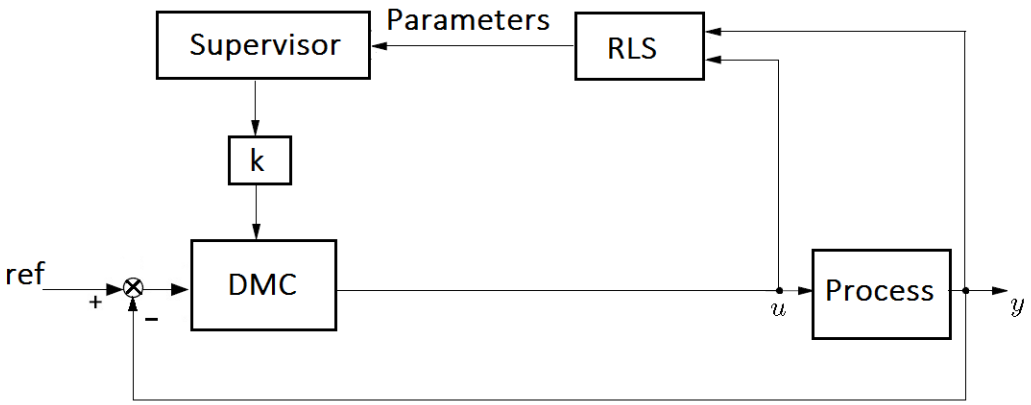

FIGURE 4.5: Proposed control scheme

Simulations are carried out with three controllers, before evaluations are made. In this chapter, another adaptive controller is introduced, so that evaluation of ADMC is not against non-adaptive controllers only.

4.1.2 Model Reference Adaptive Control (MRAC)

Already, because the PID controller is the default controller on the quadrotor available, evaluations must include PID. The PID is however tuned for the hover operating point and is not expected to perform to the quality of ADMC. However, if the PID controller were to be adaptive, i.e. adjust according to operating conditions, this adaptive PID controller must perform, perhaps even to the level of quality of ADMC. The three controllers evaluated are therefore PID, ADMC and adaptive PID realised in the form of Model Reference Adaptive Control (MRAC).

The idea behind MRAC is that, from the internal prediction model, the output of the prediction model and that of the plant are compared. The difference between the two (sometimes referred to as plant-model error) is then manipulated for control [3]. Figure 4.6 shows a scheme for MRAC.

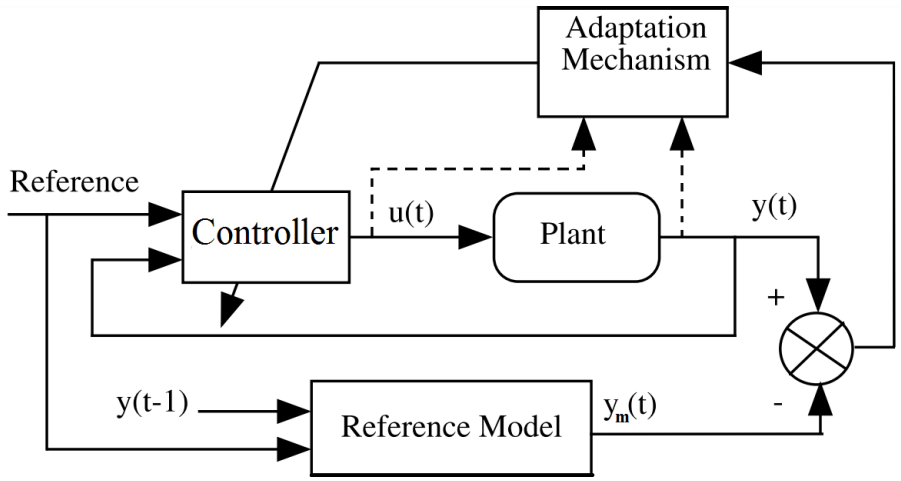

FIGURE 4.6: MRAC scheme (adopted from (3))

The scheme is very similar to that already presented in the previous chapter. The structure is flexible especially with respect to the dashed lines feeding into the adaptation mechanism. These can be neglected, depending on the desired system structure. Further explanation, equation derivation and control tuning for MRAC are available in most adaptive control literature such as [3]. This information will not be presented in this report.

The reference model used for the MRAC is the same ARX model identified earlier, while the plant is that presented in the second chapter. The adjustable controller is a PID controller, with a value of gain that is adjusted based on the plant-model error (difference between the output of the prediction model and that of the plant). The work of Ghaffar et al. is a useful piece of literature in implementation of MRAC.

Figure 4.7 shows the MRAC simulation model.

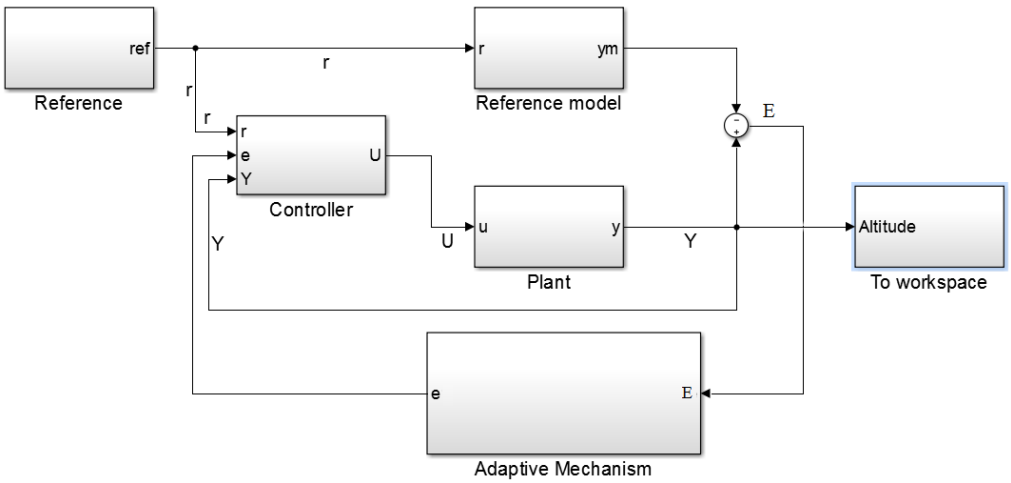

FIGURE 4.7: MRAC simulation

Figure 4.8 shows the adaptation mechanism. This mechanism manipulates the plant-model error by scaling the error. This scaling factor (C) allows adjustment of the convergence of the measured system output to the reference input, hence useful for MRAC tuning in this case, as illustrated by the results of the simulations run next.

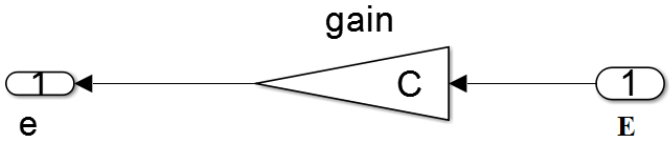

FIGURE 4.8: Adaptation mechanism

Figure 4.9 shows the controller of the model, illustrating how the PID controller is adapted, based on the plant-model error.

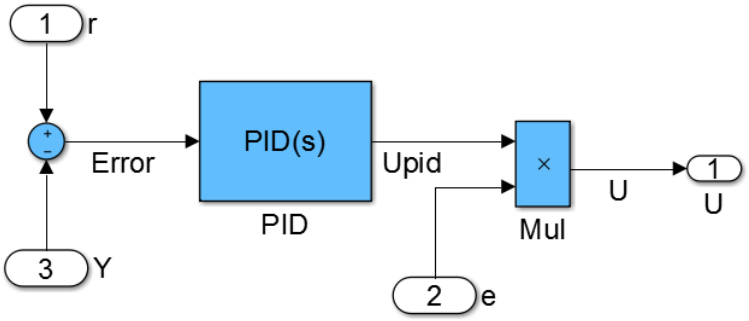

FIGURE 4.9: Controller

The overall control law (for control action U) in this case is therefore:

$$U = e * U_{PID} \tag{4.2}$$

where U_{PID} is given by 2.46.

MRAC tuning

The MRAC controller is tuned and this tuning applies to the scaling factor C. The PID part is maintained (as from chapter 2). Simulations are then run for different values of C as depicted by the step responses in Figure 4.10.

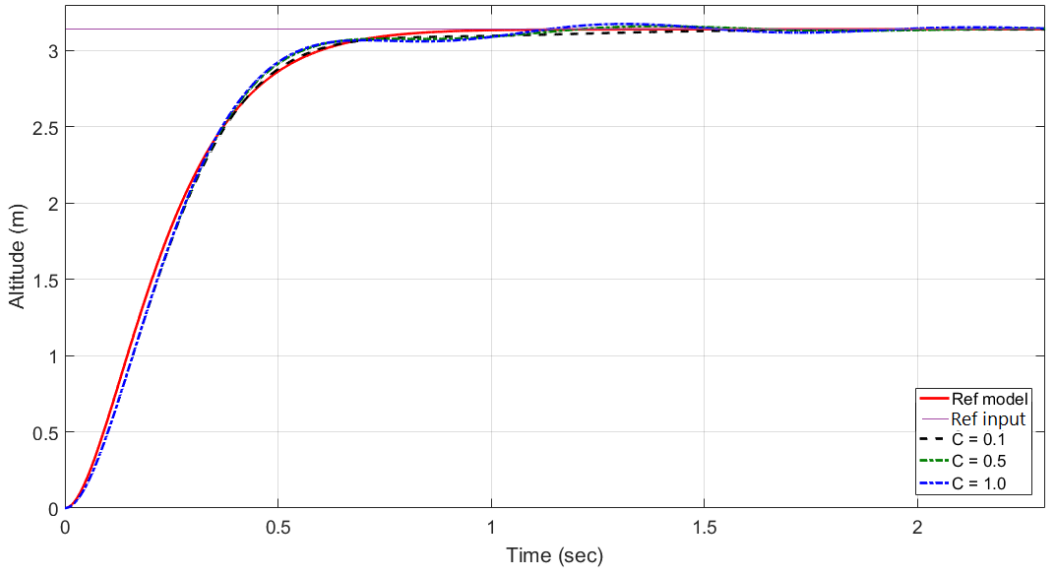

FIGURE 4.10: MRAC tuning

Figure 4.11 zooms into the results shown in Figure 4.10 for comparison.

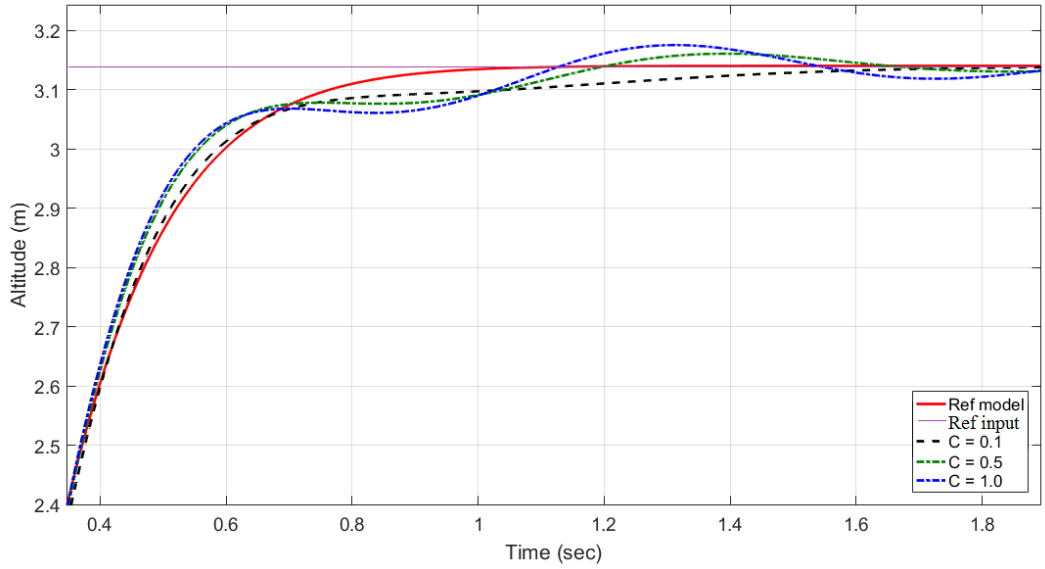

FIGURE 4.11: MRAC tuning results

When C increases, the system adaptation gains aggression, leading to oscillation as C becomes too high. The value of C used is 0.1.

4.1.3 Evaluation simulations

PID, MRAC and ADMC are simulated for the pick-and-place mission. At this stage, the purpose of simulation is to evaluate the overall performance of all concerned

controllers. The PID controller is auto-tuned to provide high quality, as much as possible. The results are shown in Figure 4.12, Figure 4.13 and Figure 4.14, which include disturbance, modelled as change in sensor output, at this point, while simulations improve gradually.

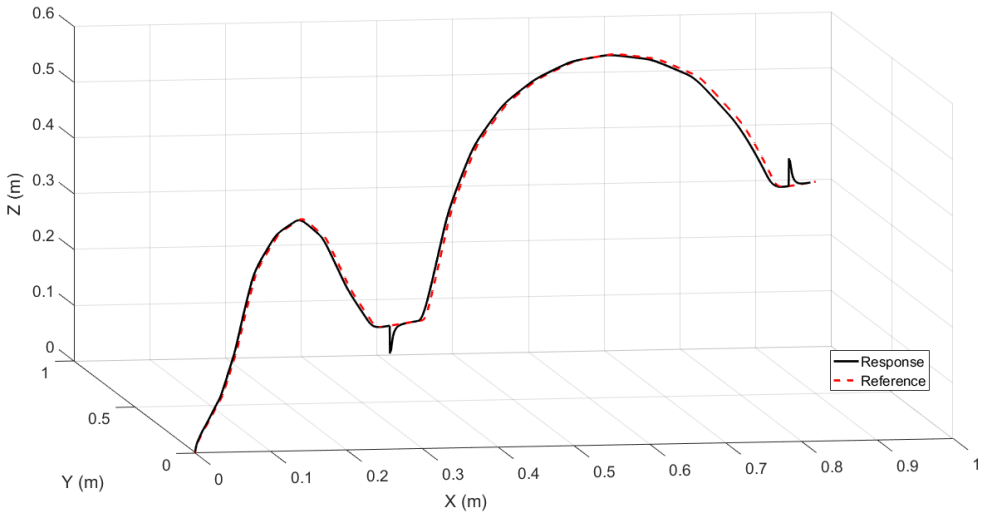

FIGURE 4.12: Pick and place - PID

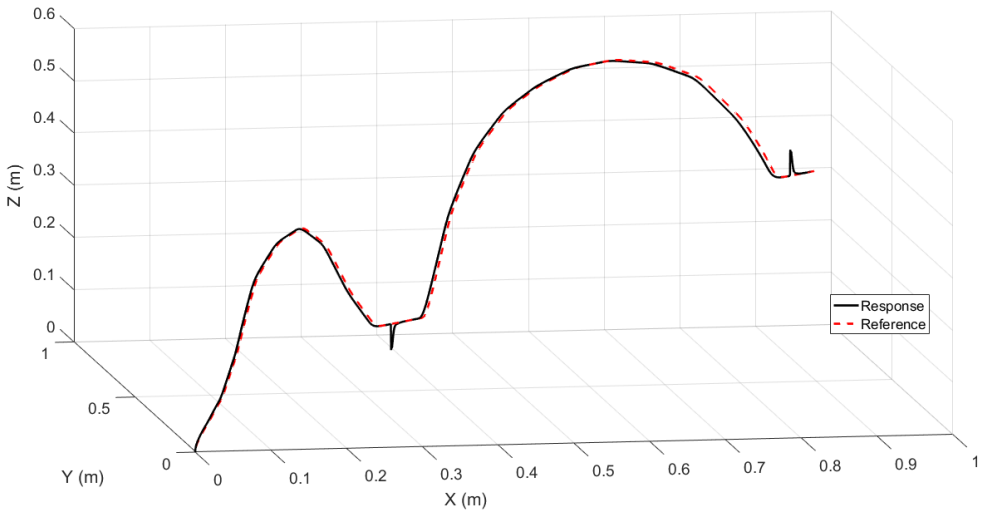

FIGURE 4.13: Pick and place - MRAC

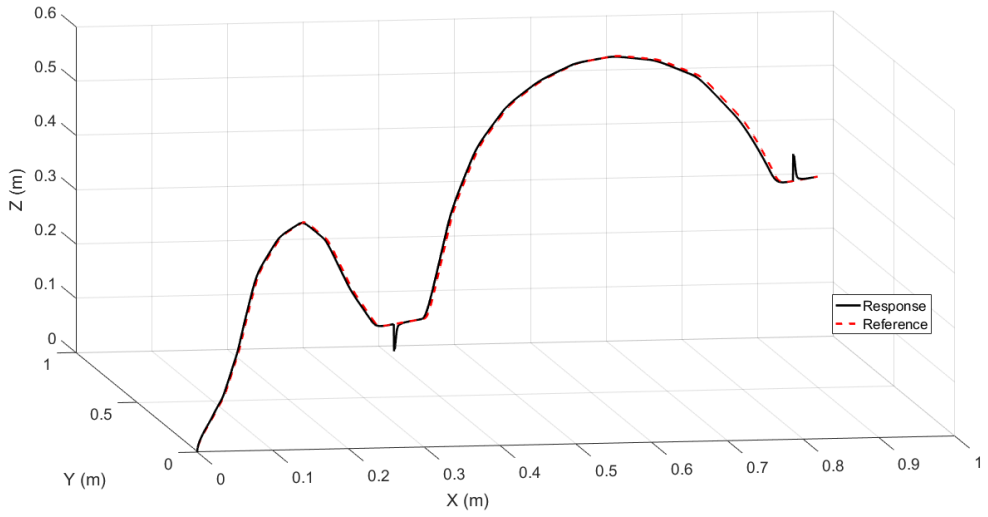

FIGURE 4.14: Pick and place - ADMC

It can be observed, in tracking how closely the response follows the reference (Figure 4.12, Figure 4.13 and Figure 4.14), that ADMC offers slightly higher precision trajectory tracking compared to the two counterparts. In terms of compensation for load pick-up and drop off, ADMC and MRAC demonstrate equal competence, while PID shows a slightly longer disturbance-correction time. Modelling the disturbance as a change in sensor output does not however give a convenient platform for the investigations of interest. Disturbance will therefore be modelled as a force for further investigation.

The experiments carried out with the quadrotor are simplified. The quadrotor is programmed to hover with altitude hold. While at hover, a load mass is dropped off the support of the quadrotor and the altitude response is recorded. Effects of the load mass can then be analysed.

4.1.4 Implementation

To achieve better simulation flexibility i.e. in disturbance simulation and reference signals, the available Matlab file model is redone as a Simulink Model. The next step moving on, is therefore to develop a Simulink model of the ADMC system. This means that a block is required to represent the ADMC algorithm as illustrated by Figure 4.15, for the altitude control sub-system, where the controller is represented by Figure 4.16.

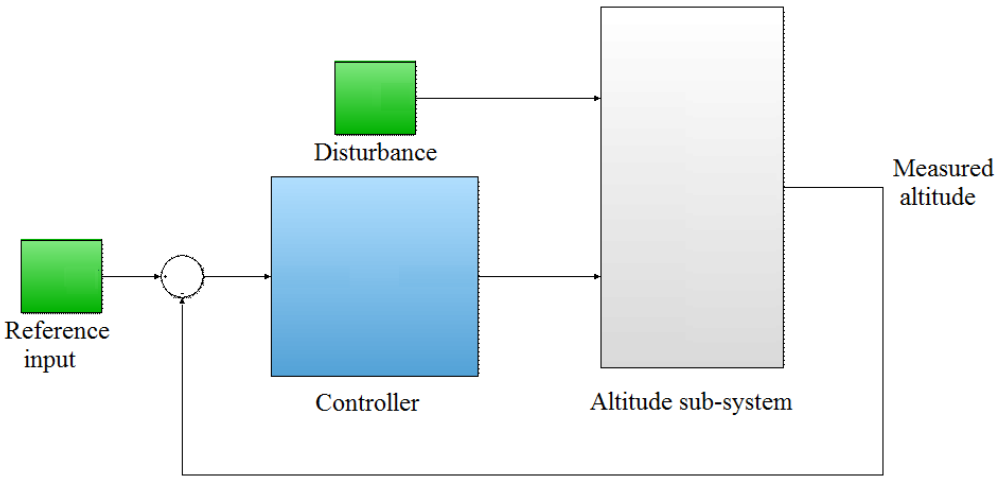

FIGURE 4.15: Functional diagram of the Simulink model

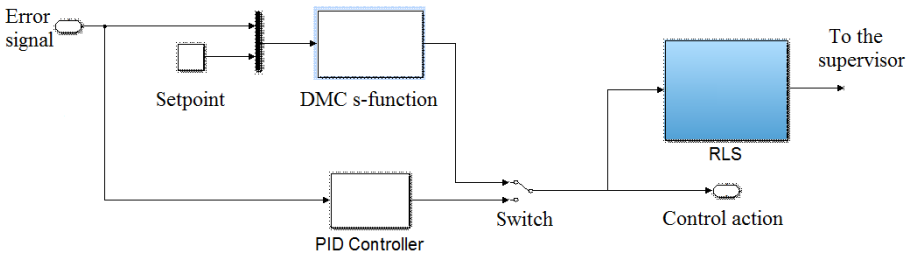

FIGURE 4.16: ADMC controller

To achieve this, it is necessary to adopt an S-function in Simulink to run the DMC script. The S-function therefore is used to describe the function block of the controller, which is in fact written in C-code. The idea then, is to set up the S-function accordingly. This involves defining functions to initialise or terminate the block and also to process inputs to outputs.

Something to note, is the presentation of the plant in the Simulink model. At this level, this research adopts an FOPTD model. The reason behind this is that it makes sense to consider command lags, measurement lags and computation or communication lags when the practical system is considered. While the consideration of the time delay is beneficial in that sense, it may drop the response time. Regardless, this even further allows for better understanding of the practical system.

Again, simulations are carried out with three controllers, PID, MRAC and ADMC as shown in Figure 4.17. The purpose of these simulations is to evaluate how each controller handles load drop-off and pick up, on a more qualitative basis.

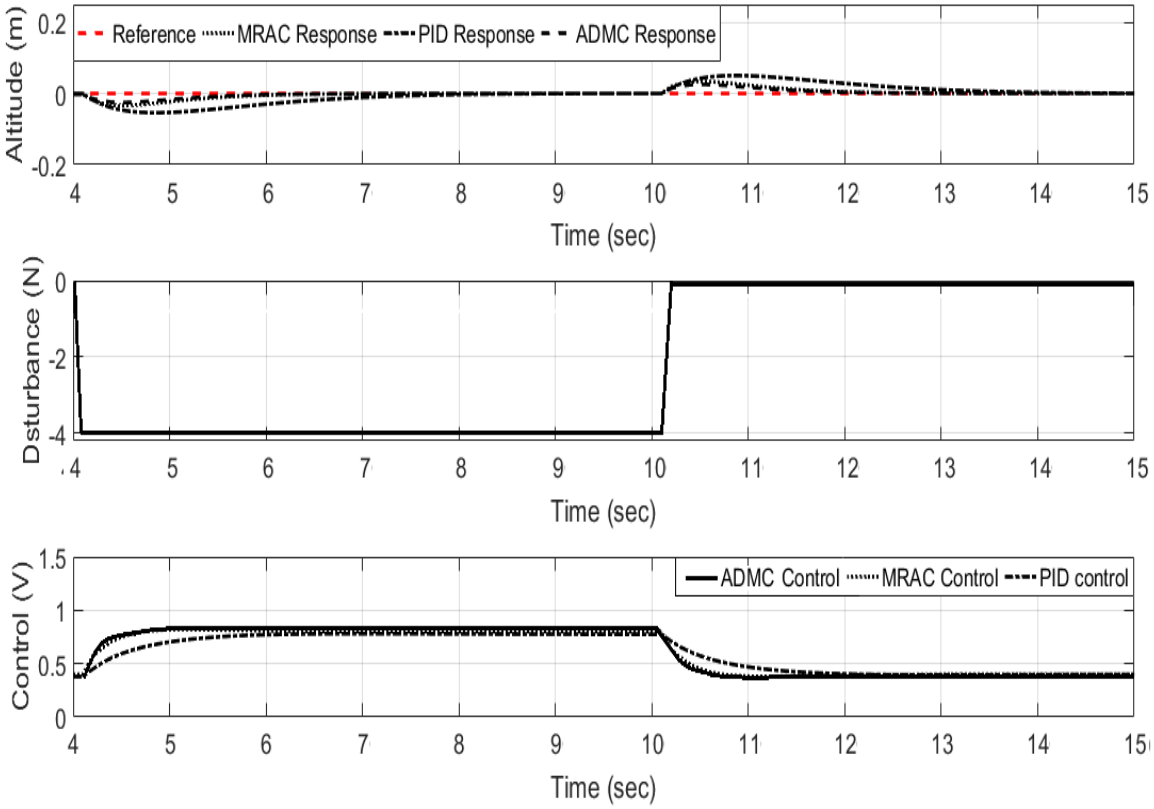

FIGURE 4.17: Simulink model simulation

These simulations show that ADMC performs better in compensating for the load mass in terms of correction response time, than MRAC or PID. This is further illustrated by the plot of the manipulated variable (control action) which is most optimal for the ADMC case. MRAC performs almost to the match of ADMC, but slightly having more disturbance error, due to the fact that ADMC incorporates an aspect of optimal control, through the DMC algorithm. PID overall, though able to provide zero steady-state error, comes with more error in magnitude at disturbance application. The error resulting from disturbance is higher in magnitude and the time taken to correct for this error. This makes sense, given that the PID controller is tuned for the hover operating point.

From the simulations carried out, it can be concluded that the adaptive controllers offer improved control to PID. Before moving to the experiments, the effect of sampling time is discussed.

4.1.5 Sampling Time

The ADMC Simulink model simulations carried out are run well with a sampling interval of 0.025 seconds. It is observed that the setting of the sampling interval has significant effect on the performance of the controller. To this end, a series of simulations is run for different sampling intervals (0.0001 seconds, 0.025 seconds

and 0.1 seconds) giving results as shown in Figure 4.18, Figure 4.19 and Figure 4.20 based on the Simulink model.

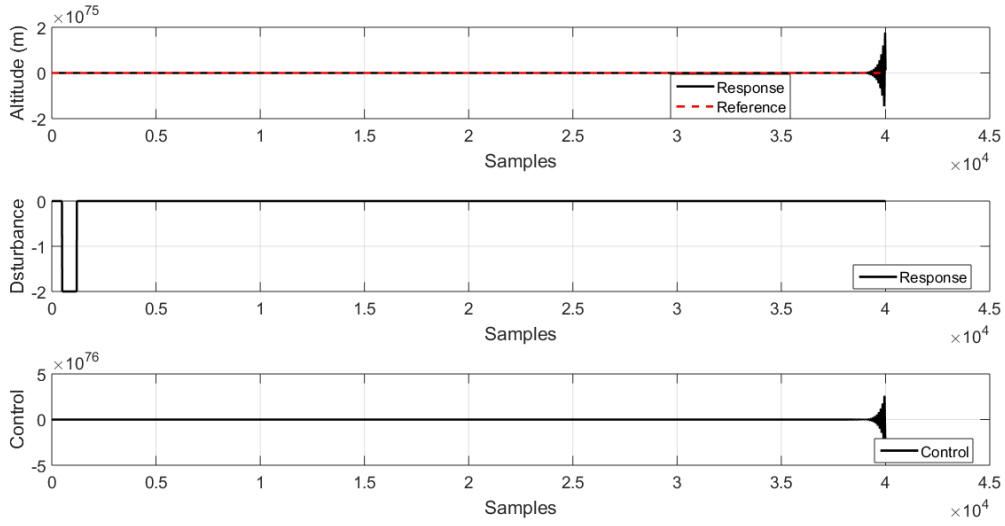

FIGURE 4.18: Sampling time 0.0001 seconds

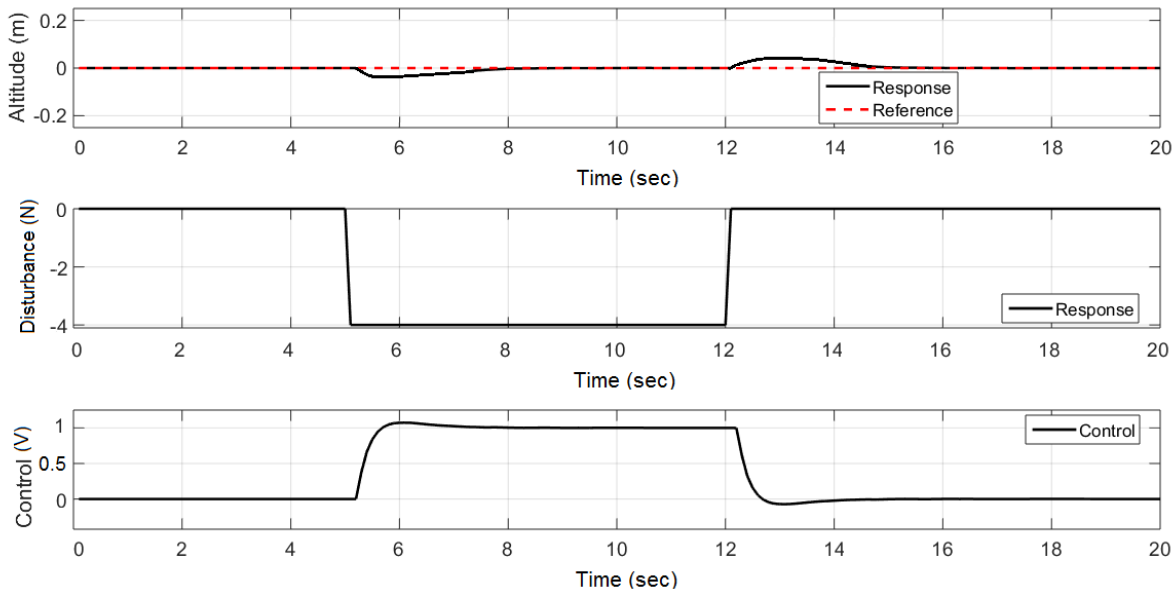

FIGURE 4.19: Sampling time 0.025 seconds

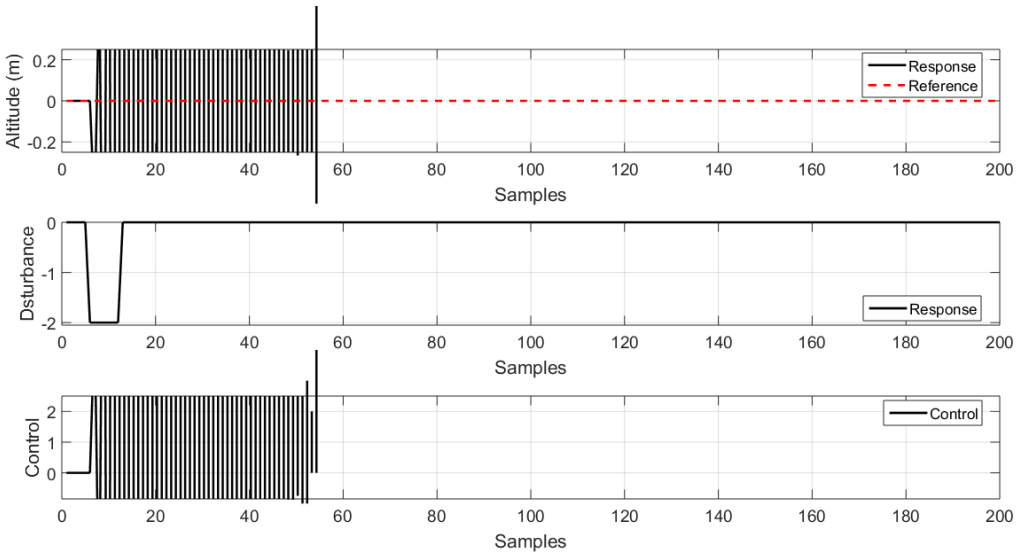

FIGURE 4.20: Sampling time 0.1 seconds

It can be concluded (and noted for the purposes of the real controller) that too large a sampling interval prevents adequate capture of the plant dynamics, while too small a sampling interval fails to allow for the full computation of the algorithm.

4.2 The Physical Experiments

As stated already, MPC is computation-intensive as the MPC algorithm solves the optimisation problem at each sample time. This optimisation problem is typically reduced to the Quadratic Programming (QP) problem. Applying DMC to real control systems is not an easy task [100]. One of the reasons for this is that the C-code required to solve the QP problem is complex and computationally expensive [101]. As a result, research into implementation of DMC on real systems, on actual microcontrollers has not received as much attention [102]. Off-the-shelf solvers do exist however, such as the qpOASES solver, and are typically used in the implementation of MPC [101] especially in industrial systems. This simplifies the implementation but interfacing the solver to the input-output modules of the system still poses a complex task.

Over time, advances in control and in computer science have constantly improved not only the theory of control, but implementation too. As such, implementation of control has found new methods, particularly, the so-called Control Prototyping approach which is based on common programming languages, typically Matlab/Simulink. In this way, it is possible to automatically generate C-code from a system model in Matlab/Simulink for example. This has been successfully done in [100, 102]. Control Prototyping, better known as Rapid Control Prototyping (RCP) is a process of deploying a controller to a real system by translating for example,

Matlab/Simulink models, to a control processing computer with input-output interfaces.

RCP can be adopted for this research in which case, the code generation can be done from a Matlab file or Simulink model. The generated code can then be deployed on a micro-controller such as Arduino. This only works if the available micro-controller is able to meet the computational demand. Alternatively, a fusion of a micro-controller and more powerful computer can be used. The more powerful computer then runs the expensive algorithm in Matlab/Simulink and sends control commands to the micro-controller. This second approach has strong benefit especially in terms of ease of data capture and signal analysis. While both approaches make controller deployment easier, it is not as straight forward. Additionally, it will be necessary to perform some manual integration to some 'legacy' code required for proper functioning of the quadrotor. This involves for example IMU sensor initialisation, processor scheduling, calibration etc., and would fall out of the scope of this research.

Typically, as in process control, the more powerful computer is the control PC, from which the human counterpart can receive data (state information) from the system and also send data (control data) to the system [100]. This works well for stationary systems where data transmission can be done via wire. In this work, ADMC must be implemented on a stand-alone computer on-board the quadrotor and tested through a set experiment. The flight controller is desired to function as follows:

1. Measure relevant information using a data acquisition system on the flight controller, with relevant sensors.
2. Send the sensor information to the on-board computer.
3. ADMC control action is calculated.
4. The flight controller sends the control action commands to the speed controllers of the motors of the quadrotor.

Two different quadrotors will be explored for experimentation. The first preliminary experiments are based on a small scale quadrotor. Findings from these experiments are then used in setting up the second preliminary experiments which use a larger quadrotor. Improvements are made to the system and then the final experiments are carried out, with the larger quadrotor.

4.3 First preliminary experiments

Throughout this work, the experiments conducted are gradually developed step by step until safety and reliability are guaranteed. This minimises any risk in the exercises, and also allows for effective planing.

4.3.1 First phase

The Parrot AR Drone is used for the first preliminary experimental testing. The AR drone is a light weight, wifi controlled quadrotor whose propeller guard frame makes it relatively safer to operate indoor. The AR drone used is shown in Figure 4.21.

FIGURE 4.21: Parrot AR drone

Library blocks are set up to interface the input and output modules of the AR drone system. The AR is then controlled from a computer running Matlab via wifi. By default, the User Datagram Protocol (UDP) is used, where speed guarantee is relatively high. Control algorithms are run on the computer and only motor commands are sent to the AR drone. This allows running computationally expensive algorithms with ease using the processing power of the Matlab-installed computer. In order to do this, a support package for the AR drone is used. Control is PID regulated, tuned optimally for the robot. Figure 4.22 illustrates the desired function as stated above.

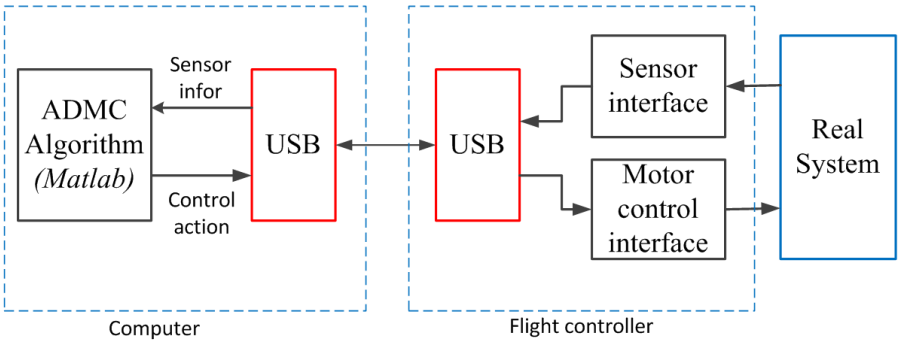

FIGURE 4.22: Functional diagram of the experimental system

To carry out the required experiments and acquire useful data, a peripheral data acquisition system is proposed. This isolates the data measurement from the control computation during flight. Measured data is not processed in real-time, but is stored for post processing and analysis. Processing power is prioritised for the control computation. This peripheral system comprises an Arduino computer (microcontroller), ultrasound sensor and bluetooth module as shown in Figure 4.23.

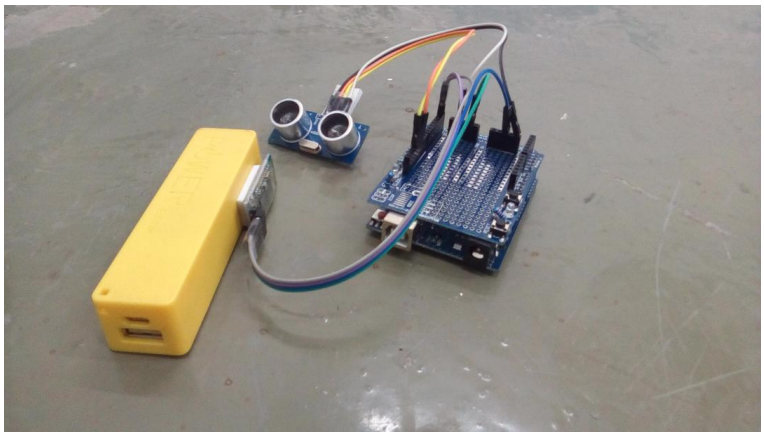

FIGURE 4.23: Peripheral data acquisition system

Experiments are carried out to investigate the general performance of ADMC and PID control. The quadrotor is set to hover at 0.75 m above the floor, which is a safe height in case of unexpected emergencies and, given the small size of the drone, the ground effect turbulence is negligible at this level. The settings of ADMC are as shown in Table 4.1.

TABLE 4.1: ADMC parameters

Parameter	**Value**
Prediction horizon (p)	5
Control horizon (c)	5
Weighing factor (r)	0.5
Aggression factor (α)	0.7

First, with the added weight of the peripheral system (which required a separate power source), the quadrotor is unfortunately unable to gain sufficient lift to rise to the desired height above the floor. The quadrotor is only able to hover less than 15 cm from the floor. This however presented an opportunity for a quick analysis of the general performance of ADMC and PID control on a real system. The behaviour of the two controllers are illustrated in Figure 4.24 and in Figure 4.25.

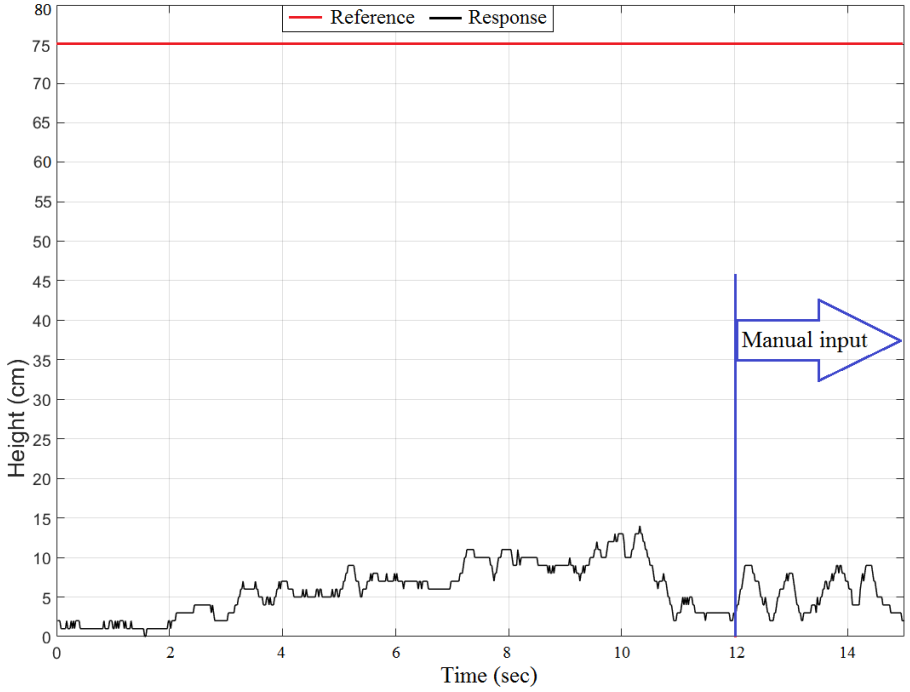

FIGURE 4.24: PID low level hover

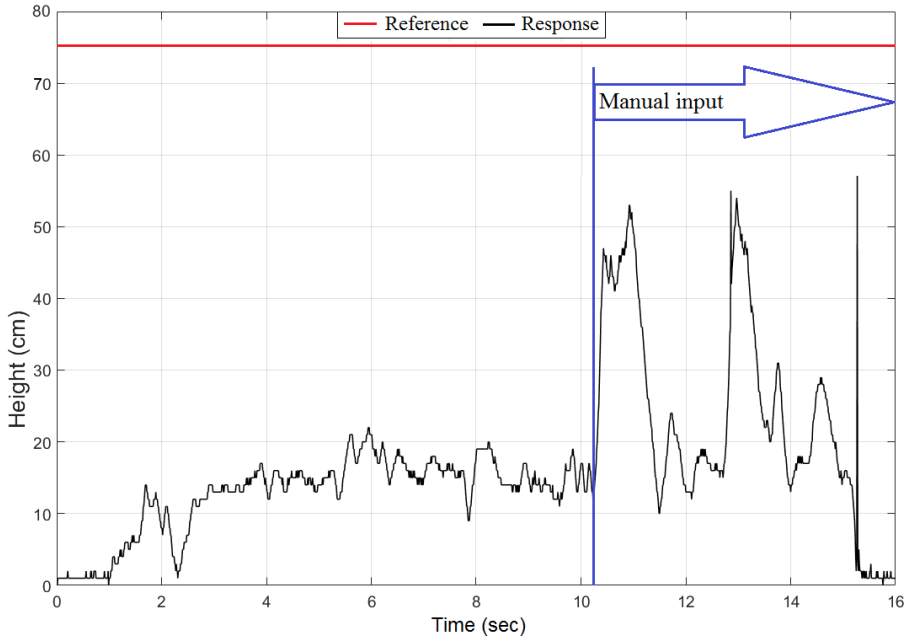

FIGURE 4.25: ADMC low level hover

First phase observations

It is observed that ADMC is more aggressive in attempting to track the set 75 cm and does actually get closer to the target, than the Parrot AR PID controller. This

is due to the relatively high value of the aggression factor (0.7). Given the lift deficiency, throttle is stepped up manually at the points indicated by blue arrows, to force the quadrotor to gain lift. Again, with ADMC, this has an effect of raising the quadrotor in height. The PID controller does not respond to the increased reference height. This proves the ADMC aggression in control which in this case results in more aggressive command tracking.

4.3.2 Second phase

Having adjusted weighting on the quadrotor by removing the propeller guard, experiments are repeated. This time, the lift generated by the motors is enough to raise the quadrotor to the set 75 cm. Results are illustrated by Figure 4.26 and Figure 4.27. In these experiments, the load disturbance is applied as marked by the red circles, by a matter of forcing the quadrotor slightly below the set hover height and then releasing the force for the quadrotor to correct back to the reference altitude.

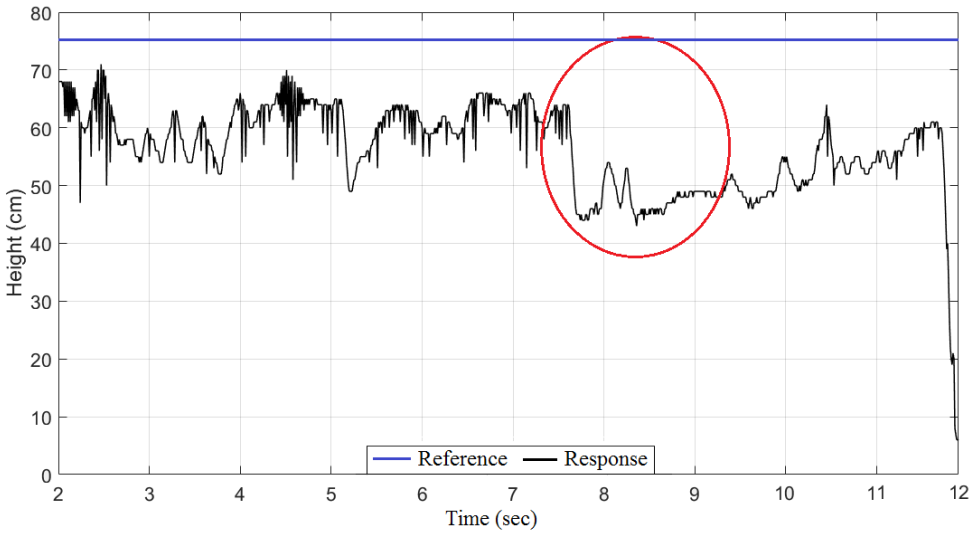

FIGURE 4.26: PID response

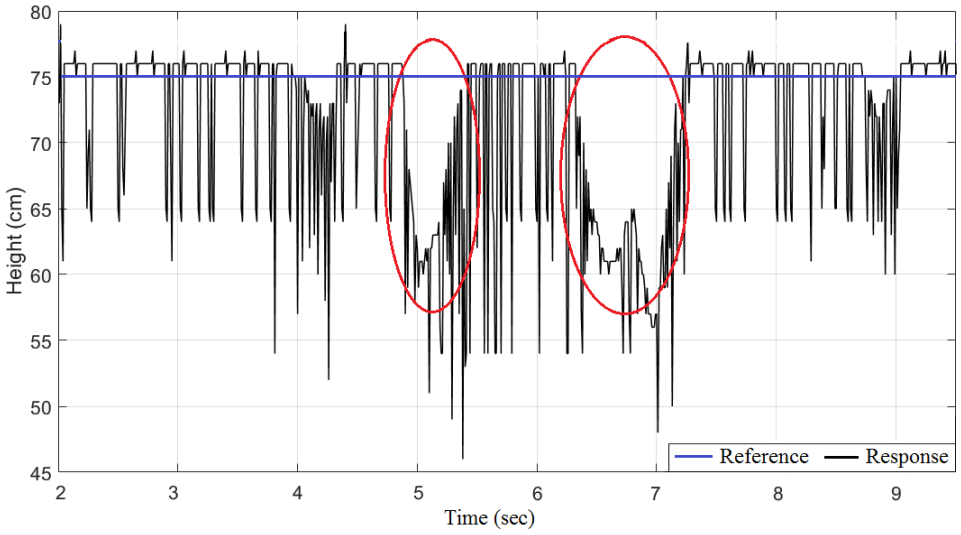

FIGURE 4.27: ADMC response

Second phase observations

Analysing the results, ADMC out performs PID in both maintaining steady height and in correcting for the load disturbance. The noisy oscillation from Figure 4.27 is a result of tilt in pitch and row as the quadrotor attempts to maintain stability. The aggressive ADMC corrections result in more defined tilt and this is picked up by the peripheral ultrasound sensor as measured distance changes, and rapidly. The off-set of ADMC is a matter of different calibration for the peripheral ultrasound sensor and that of the Parrot drone, and is at this point not important.

4.3.3 Unexpected ADMC flaws

A problem identified is that while the ADMC setting results in unnecessarily aggressive (rough) control, reducing the aggression factor slightly (from α = 0.7 to α = 0.6) however lowers the ADMC correction response-time to almost the same level as PID. This means that the ADMC expensive algorithm cannot be justified, unless some improvement is implemented for ADMC. A possible and simple improvement would be to determine the value between α = 0.7 and α = 0.6, which gives the best response. The approach taken in this work is to employ some algorithm to smooth out ADMC, as will be detailed later.

4.4 Second preliminary experiments

The next preliminary experiments are based on tests conducted outdoor, with the Axe quadrotor, illustrated in Figure 4.28 below. These experiments are an improvement from the tests already presented.

FIGURE 4.28: Axe quadrotor outdoor

4.4.1 Articulation of the experimental process

With the Axe, all computation is carried out by the flight controller on-board. This quadrotor has a larger payload capacity, maximum of approximately 0.7 kg. The experiment is set up such that ADMC is applied to the altitude control system while the attitude subsystems are PID regulated, in a coupled set-up, as presented by the system model in chapter 2. The reasons for this are as follows:

1. To minimise the on-board computational requirement to levels that the flight controller can handle.
2. To minimise risk. The Axe is a relatively large drone. It is worthwhile to invest in reliable systems such as the Vicon motion capture system for fuller exploitation on experimental basis.
3. Exploitation of the altitude system alone is expected to produce meaningful results for informed conclusions.

The Axe is equipped with a Pixhawk flight controller. The idea is to execute the ADMC (and PID) control on this autopilot, such that the sensors available with this flight controller are used. This avoids the use of peripheral systems hence avoiding compromise in system and process integrity. To assist with this, Pixhawk Pilot Support Package (PSP) is used. PSP is useful because this package offers an efficient, safer and practical method to implement custom controllers on the Pixhawk autopilot. The method adopted will investigate PID control and ADMC and at a later stage, adaptive PID is added. PID is investigated first because PID is much simpler and less prone to error or failure resulting from complexity.

The set-up of the Axe for experiments is such that:

1. Altitude is measured by the barometric sensor on the Pixhawk flight controller. This sensor provides more stable and reliable measurement compared to the GPS altitude measurements.
2. GPS data is used to stabilise the quadrotor on the X-Y axis.

The default PSP model does not include the barometer and GPS sensor. The model is modified to utilise the barometric and GPS sensors. The final model therefore comprises the sensors shown in Figure 4.29.

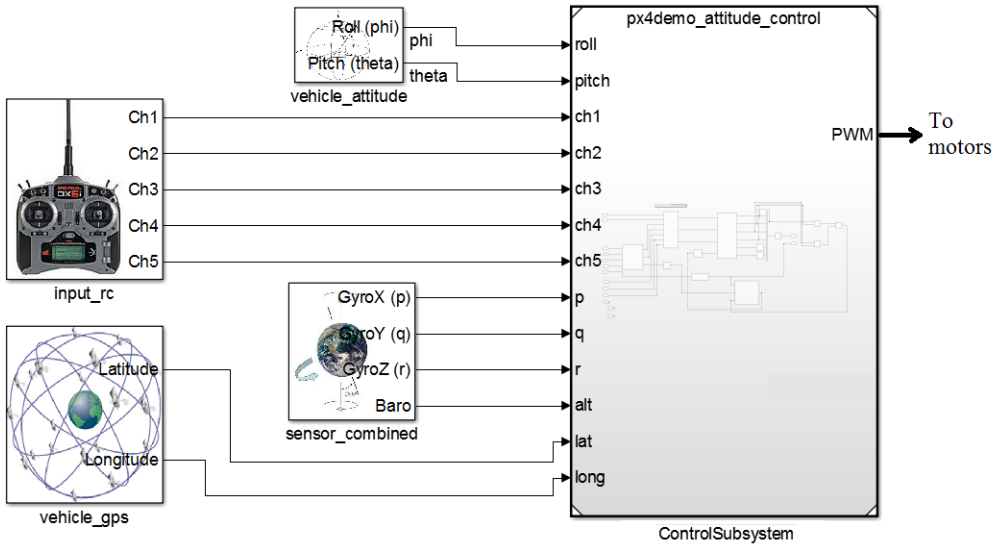

FIGURE 4.29: Modified Pixhawk quadrotor model

The source code files and header files (containing declarations) are set up, for full integration of the GPS and barometric sensor. Details regarding the code generation from the model, for Pixhawk using the Pixhawk support package are available in Appendix D and are guided by instructions given in [106].

Experiment strategy

Experiments are carried out outdoor, in GPS assisted mode of flight, for safety reasons. This ensures stability in the X-Y directions. Flight data is written onto the Pixhawk flight controller micro-SD card for analysis. The idea is to therefore hover the loaded Axe quadrotor then drop the load. The experiments are set up to evolve and gradually improve. The first and second preliminary experiments are therefore building stones for the final experiments, so it is from the final experiments that conclusions with substance will be written up.

Reference input

The reference input for each experiment will vary. The reason for this is that from take-off, the quadrotor is flown to hover with altitude hold, in the shortest time possible. This level, for each flight, is then taken as the reference altitude.

Payload release

A payload release mechanism is designed and mounted safely on the Axe. This mechanism, at this point, is remotely controlled from a unique radio, other than that used for commanding the Axe quadrotor. The Axe therefore takes off, with some load and the load is released at hover and the effects of this load drop off on altitude are analysed. The load mass is a sand bag, weighed and marked accordingly. In this way, by releasing the load, a disturbance is introduced.

Choice of results visualisation

The visualisation of experimental results must be standardised so that meaningful analysis can be executed and useful conclusions derived. The final experiments (which close this chapter) are more standardised because these experiments are improved from all prior experiments. For this reason, results visualisation is emphasised for the final experiments.

Chosen variables plotted

The variables plotted for each experiment must pronounce indices that allow accurate evaluation of the different controllers involved. Again, these variables are emphasised for the final experiments.

4.4.2 Aggression control for ADMC

As highlighted after the first preliminary experiment, the aggression of the ADMC controller needs attention. This must be done preferably, without dropping the value of the DMC aggression factor as this was observed to reduce the response time of ADMC to levels close to PID. To address the matter at hand, a smoothing function is proposed for post ADMC processing. Specifically, the Hyperbolic Tangent function (first used in the work of Sauri L in 1774) [107] is proposed. The hyperbolic tangent (tanh() function) is proposed for smoothing the aggression of ADMC. An advantage of the tanh() function is that this function is readily available in many programming languages, including Matlab. The hyperbolic tangent is a function useful in controlling the transition from one point to another [108]. The hyperbolic tangent function is defined as the ratio of the hyperbolic sine and hyperbolic cosine in radians (4.3) [107]:

$$\begin{aligned} tanhx &= \frac{sinhx}{coshx} \\ &= \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \\ &= \frac{e^{2x} - 1}{e^{2x} + 1} \end{aligned} \tag{4.3}$$

in which case $sinhx$ represents hyperbolic sine and $coshx$, the hyperbolic cosine on a radian scale. In simulation, the hyperbolic tangent can be wtitten as:

1. x = -4:0.1:4;

2. plot(x,tanh(x));
3. grid

Figure 4.30 gives a simple illustration of the hyperbolic tangent signal smoothing.

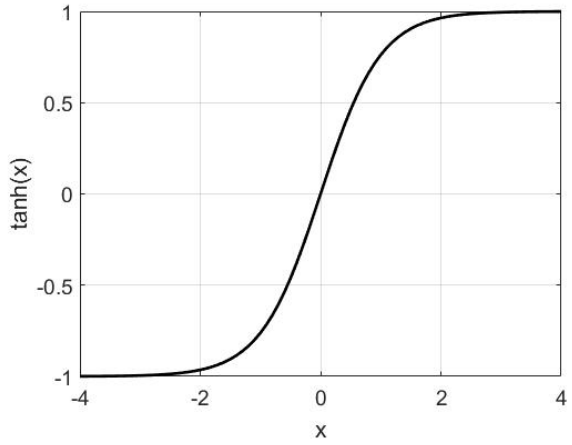

FIGURE 4.30: Hyperbolic function

For more flexibility, the hyperbolic tangent function is defined as:

$$func(x) = 2 * (0.5 + 0.5 * tanh(k * x)) \tag{4.4}$$

where k is a smoothing factor, as explained in Figure 4.31. In this way, the DMC control action is smoothed. The radian input of the hyperbolic function is mapped to the manipulated variable scale, which is in fact a pulse between 1000 micro-seconds and 2000 micro-seconds.

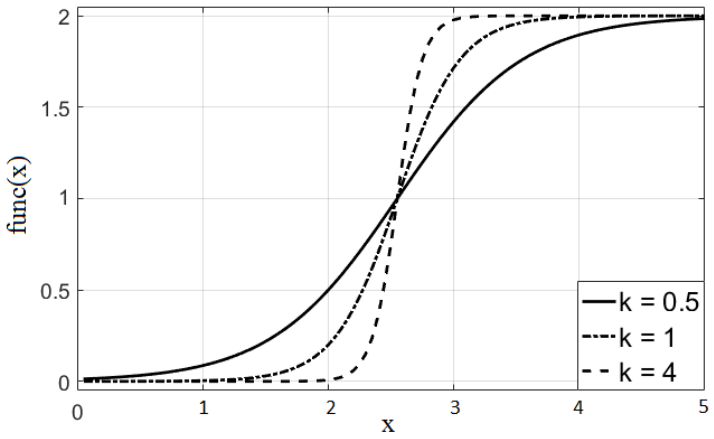

FIGURE 4.31: Hyperbolic function smoothing factor

The hyperbolic tangent smoothing allows for the aggression factor of DMC to be stepped up from 0.7 to 0.82. Other changes made, in the interest of reducing computational load, include reduction of the step response model to size 8 (with steady state from the third point), prediction horizon to 4 and the control horizon to 2. The Axe experiments are carried out with this setting, as summarised by Table 4.2.

TABLE 4.2: Adjusted ADMC parameters

Parameter	Value
Prediction horizon (p)	4
Control horizon (c)	2
Weighing factor (r)	0.5
Aggression factor (α)	0.82

4.4.3 Results of the second preliminary experiment

The flight logs of the PID and ADMC controllers are analysed. It is important to note that, while the Axe quadrotor has meaningful payload capacity, the Axe does present some issues with respect to safety. The relatively large size of the Axe means higher operational risk, especially in experimentation. For this reason, experiments are minimised to only demonstrate the necessary aspects, for response comparison. Experiments are conducted as follows:

1. Implementation of ADMC on the altitude control system. The idea is to run experiments for the PID, standard ADMC and smoothed ADMC.
2. Experiments are improved step by step until it is possible to draw meaningful conclusions.

Several flights are carried out under calm wind conditions and flight logs stored on the SD card are closely analysed and presented, with the aid of relevant software. The PID controller is investigated first.

PID

Four load masses are available for tests, 0.15 kg, 0.3 kg, 0.4 kg and 0.6 kg. The 0.15 kg is experimented with first. When released, the load mass of 0.15 kg is observed to not cause sufficient and measurable disturbance to the system. The 0.6 kg load, on the other hand, is rather too heavy for the quadrotor, causing the quadrotor to struggle, especially at take-off. For this reason, the 0.4 kg and 0.3 kg load masses are adopted and used for the experimentation. These are able to induce measurable disturbance, and also ensure safety in carrying out experiments.

Figure 4.32 and Figure 4.33 show the response of the PID controller when the load mass of 0.4 kg is dropped. The altitude given is as measured by the on-board barometer and the red arrows mark the point of load drop-off, this applies to all experiments (second preliminary experiment). This load drop-off point is determined by analysing the response of the quadrotor against the input of the pilot (in landing the

quadrotor), which gives an indication that the experiment is complete and the load was released. This strategy will be improved in the final experiment. Experimental data in this section is presented only for and around the point of application of disturbance because it takes fairly long to actually get the quadrotor to hover stably (because of the load) before triggering the load drop-off. This is something that must and will be improved for the final experiment, before concluding this work, but for the purposes of preliminary tests, data is analysed.

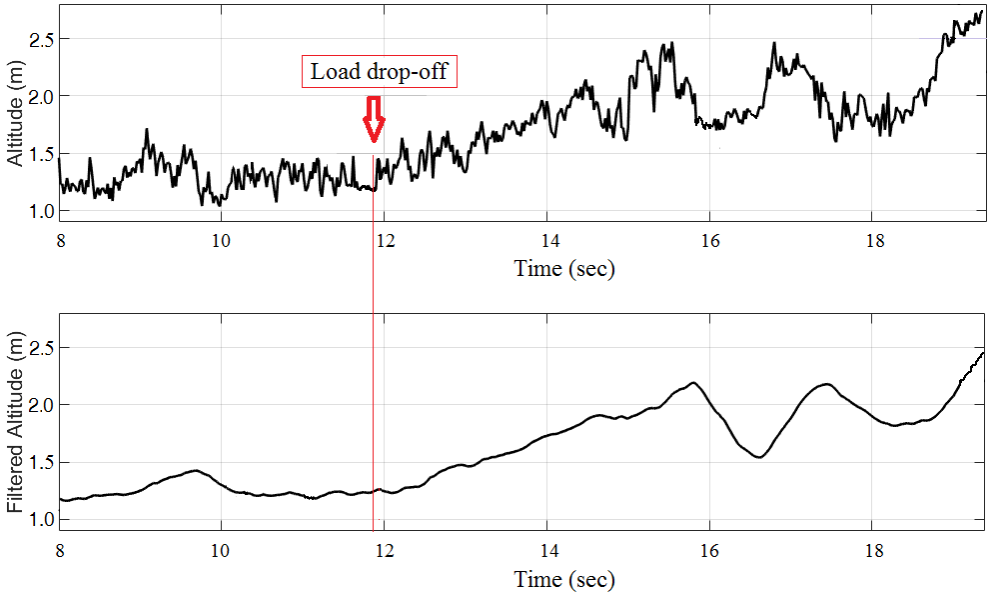

FIGURE 4.32: First PID experiment

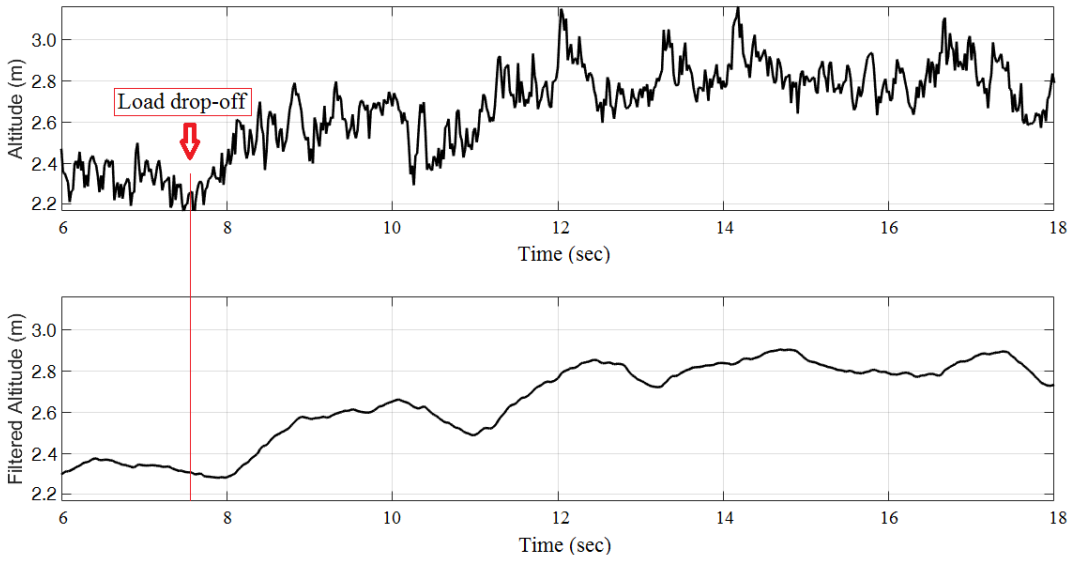

FIGURE 4.33: Second PID experiment

Immediate observations

The first observation is that altitude rises at load drop-off, without the compensation that is expected, as with simulation. To analyse this case, an understanding of the architecture of the system algorithm is required. The algorithm implementing the altitude-hold relies on sensors that are affected by some practical factors. While altitude hold must maintain altitude accurately, magnetic interference on the compass and vibrations compromise the accuracy and steadiness of the output. Experimental data cannot therefore be as clean as simulation data but, because the same quadrotor is used for all experiments with the different controllers, any superiority or deficiency of a particular controller will self define.

Another observation worth pointing out is that the measured data is in fact quite unstable (noisy). Making this worse, is the fact that the noise is rather inconsistent from one experiment to the other, one minute to the next. The reason behind this is that the barometric sensor is affected by atmospheric conditions such as pressure change and wind.

PID observations

To allow for comparison of the different controllers, performance indices are used. At this point (preliminary experiments), the indices used are stability and error. In control engineering, a control system is generally stable if every bounded input to the system gives bounded output response. It is observed that the PID controller manages to maintain stability in altitude after load drop-off. The filtered plot of the raw data is meant to highlight the trend in the data. The maximum altitude error resulting from load drop-off can be approximated to 1.3 m.

Standard ADMC

The standard ADMC (without smoothing) is investigated next. Based on the first preliminary experiment, standard ADMC is expected to be relatively aggressive in correcting for disturbance and also in reference tracking. Experiments with this controller are conducted with care. The first flight shows some uncontrollability even to the point where it is impossible to execute the actual load drop-off. The cause of this, is the combined effect of high aggression and also unstable sensor output from the measurement system. The plot of 'Altitude PWM' on Figure 4.34 shows the applied control command input in stabilising the quadrotor. This, in reality, is the pulse width of the signal, which ranges from a minimum of approximately 1000 microseconds to a maximum of approximately 2000 microseconds, where 1500 microseconds should typically cause the quadrotor to hover. Slight command input adjustments are observed to result in excessive changes in altitude, in some cases causing rapid and unsafe drop in altitude (circled in red) on Figure 4.34 below.

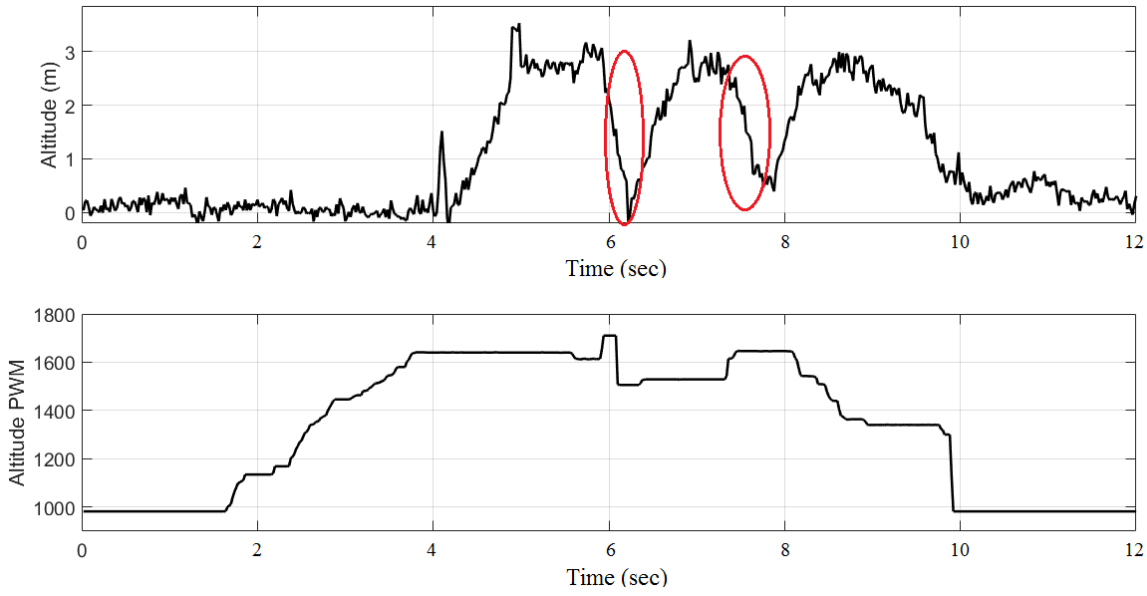

FIGURE 4.34: First standard ADMC experiment

Standard ADMC observations

It is concluded that the standard ADMC is not entirely safe for immediate deployment on the physical Axe quadrotor. The aggression with the Axe quadrotor is in fact observed to be relatively more pronounced than that of the smaller AR quadrotor used in the preliminary testing. From a safety perspective, further experimental testing cannot go on without improving the controller or measurement system. In this way, the smoothing algorithm is therefore added to the standard ADMC.

Smoothed ADMC

Flight tests with smoothed ADMC should reveal the influence of the smoothing algorithm implemented. Figure 4.35 and Figure 4.36 show the response of the ADMC controller with smoothing, when the 0.4 kg load is dropped. Again, experimental data is presented only for and around the point of application of disturbance.

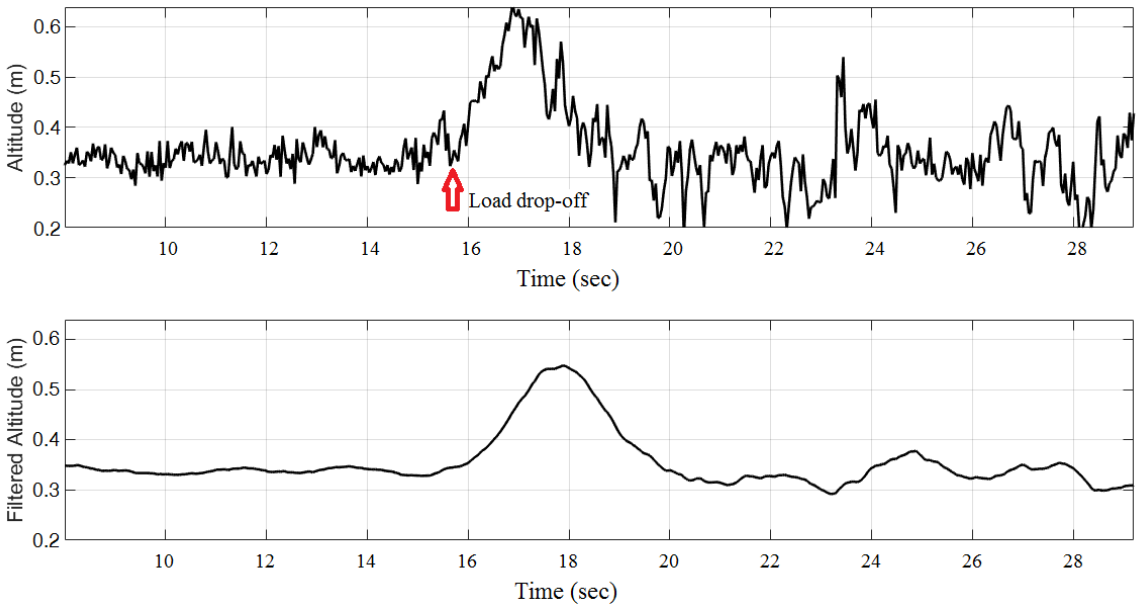

FIGURE 4.35: First smoothed ADMC experiment

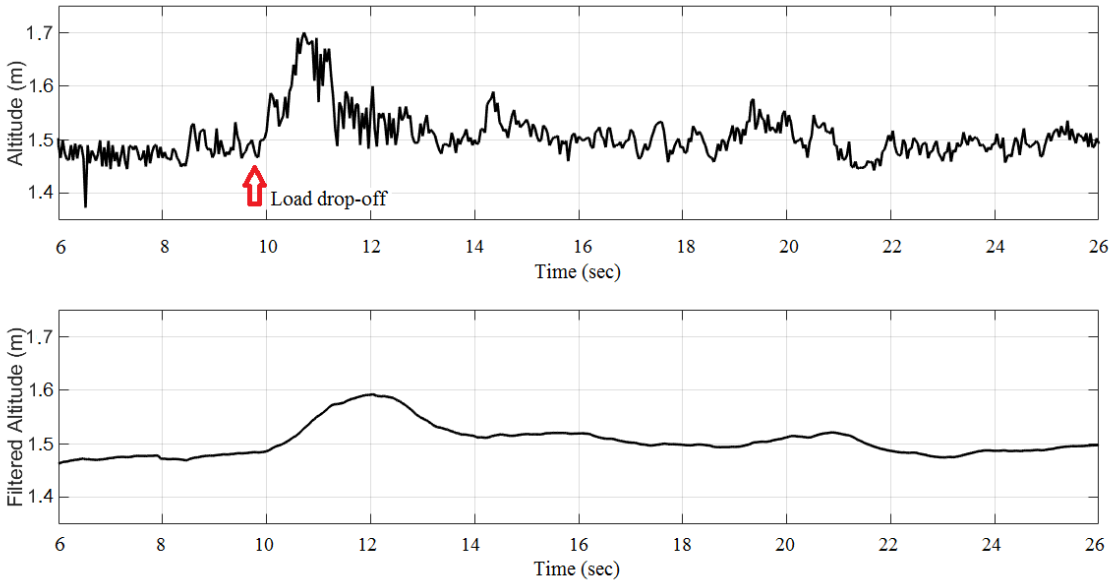

FIGURE 4.36: Second smoothed ADMC experiment

While Figure 4.35 and Figure 4.36 show improved performance for smoothed ADMC compared to the standard ADMC (Figure 4.34, it is important to however point out that one or two of the total number of flight tests with smoothed ADMC did reveal unfavourable aggression. The aggression of these discarded flights is however less pronounced than that of the standard ADMC.

Smoothed ADMC observations

Smoothed ADMC shows improved control (compared to the standard ADMC) to the point where load drop-off tests can actually be executed safely. It is observed that smoothed ADMC manages to maintain stability in altitude after load drop-off. The maximum error can be approximated to 0.3 m. Based on these accomplishments, the effort required in developing ADMC can already be justified.

General observations

Other observations from the second preliminary experiment include:

1. The output data of the barometric sensor is noise polluted, something expected in real systems. In particular, the barometric sensor is actually slightly affected by wind and pressure differences. The result is that the noise in the measured data will not be consistent from experiment to experiment.
2. The method of determining the point of disturbance application must be improved. Preferably, the available system must cater for disturbance triggering, instead of introducing an external system.
3. While smoothed ADMC reveals improved control for most of the flight tests, the few flight test that revealed undesirable aggression and control difficulty are a cause for concern. The final controller must be designed to work as desired at all times with no chances of misbehaving. The general ADMC code must therefore be revisited and analysed.
4. The simulation-tuned ADMC controller cannot be implemented without significant adjustment. Experimentation does introduce deeper understanding of the controller and how best to implement the controller.
5. By readjusting PID gains for different operating points, PID control may gain more quality, for the particular operating point tuned for. This comes back down to adaptation, in which case, adaptation is useful because adaptation eliminates the need for controller re-adjusting for each different operation. As with simulations, it is helpful to add tests which include some adaptive PID control so that experimentally, ADMC is not only evaluated against PID only, but adaptive PID too.

Adjustments to address the negative observations

Based on the observations as stated in Section 4.4.3, certain elements of the system must be adjusted. It is expected that these adjustments will results in improved performance and measure-ability.

Adjustments made to the system are as follows:

1. First, the noisy altitude data is addressed. Less noise in the data will allow better analysis of output data. The solution is to introduce a more stable sensor.

An ultrasound distance sensor is proposed. While this sensor gives more stable readings, it is not entirely noise-free especially as distance increases. This sensor has a maximum range of 6 m and so flight tests are conducted within a limit of 6 m from the ground. On the hardware side, the sensor is connected to the flight controller via the I2C connection. The sensor is mounted below the payload bay as shown in Figure 4.37.

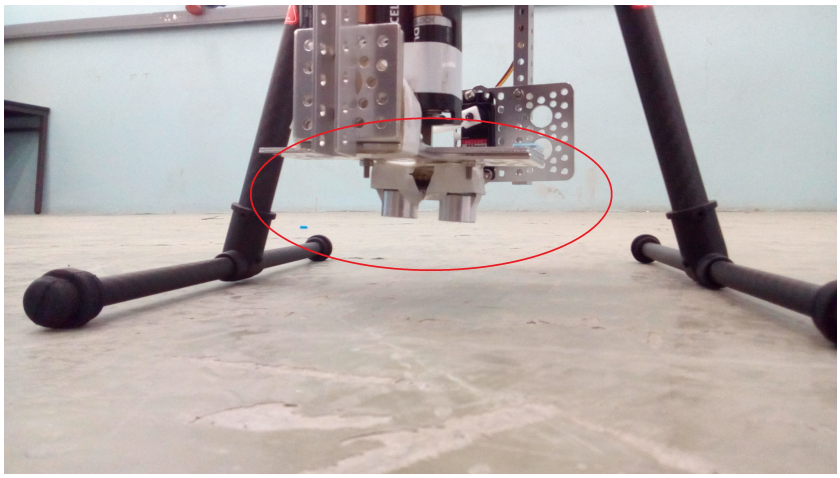

FIGURE 4.37: Ultrasound distance sensor

On the software side, it is necessary to build the Simulink block for the ultrasound sensor for the Simulink model, prior to code generation. To accomplish this, the S-Function approach is adopted.

2. To ensure precise tracking of the disturbance application time point, the load triggering system is centralised to the flight control system. A button is programmed on the remote controller to trigger the load drop-off. This button, like all other buttons, is monitored and logged by the flight controller and then used as a reference in determining the exact point of disturbance application.

3. To ensure that the ADMC behaves as expected (without control difficulties), the system must be analysed to determine potential causes of the seasonal difficulty in control. The problem is of the nature that, in some cases, the quadrotor (ADMC based only) tends not to respond to control input as commanded, in real-time. The documentation of the Simulink Support Package for the AR drone, used in Section 4.3 specifies that, for the AR quadrotor, incorrect sample rates will result in incorrect system behaviour. In the case of the Axe quadrotor, if the ADMC sample rate is too low (for the processor), this will result in fragmented data due to loss of data. Now, when a particular data packet is lost (dropped) at any given point, the data from the previous time point is in fact used. Modifying the sample rate (set to 40 Hz at this point) of the ADMC controller is therefore the first step taken, in trying to solve the issue stated in item 3 of Section 4.4.3. The sample rate is increased by a factor of two.

4. An adaptive PID controller is implemented for the physical system, for the purpose of adding substance to the evaluation of ADMC on an experimental scale.

4.5 Final Experiments

Flight tests are repeated for the improved system (adjusted according to Section 4.4.3), for the PID, adaptive PID, standard and smoothed ADMC controllers. Three experiments are considered for each controller, for both the 0.4 kg and 0.3 kg load.

4.5.1 Choice of results visualisation and variables plotted

The visualisation of results is reliant on the capability of the system and the available sensors. The presentation of results graphically illustrates four variables, which are, the measured altitude, filtered altitude, load PWM and pilot PWM. Collectively, these plots allow evaluation of the different controllers, also in the capacity of the system capabilities.

Load PWM

The Load PWM plot shows the point at which the load drop-off was triggered, and does so accurately, with reference to precise point in time the PWM signal was sent to trigger the load release.

Pilot PWM

The fourth graph (Pilot PWM) shows the pilot altitude input control command. This is the pulse width of the signal, which ranges from a minimum of approximately 1000 microseconds to a maximum of approximately 2000 microseconds. At approximately 1500 microseconds PWM, the quadrotor is expected to hold altitude, when unloaded. This upper limit of 2000 microseconds (which is not reached in this work) means that at this point the input signal is saturated. This translates to maximum lift.

At take-off, the quadrotor will not immediately gain altitude when the Pilot PWM reaches 1500 microseconds! This is because thrust needs to first build up, as the motors gain in rotational velocity, until enough lift force is produced.

Reference input

The reference input is indicated on the measured altitude plot from the the point of load drop-off. The explanation is that it makes sense to consider the altitude at which hover is achieved, as the reference input. The experiments are conducted in

such a manner that on releasing the load, the quadrotor is given time, in which the response is monitored, before the pilot takes control to land the quadrotor safely and fly the next mission.

Determining steady-state

An important aspect in evaluating the different controllers is an analysis of the system disturbance response. The key, is to correctly take note of how (or when) the disturbance is compensated. This requires the ability to correctly determine the point where steady-state is reached, after the effect of disturbance, despite the noise pollution affecting the measured data. A flight test with the quadrotor unloaded is used to determine the general response of the quadrotor hover steady-state. Figure 4.38 below shows the steady-state condition (approximately 18 seconds to 23 seconds) as recorded by the flight controller. During this interval the altitude reading displayed on the ground control computer is fairly constant. The acceptable margin of fluctuations due to noise is set to $\pm$ 0.15 m.

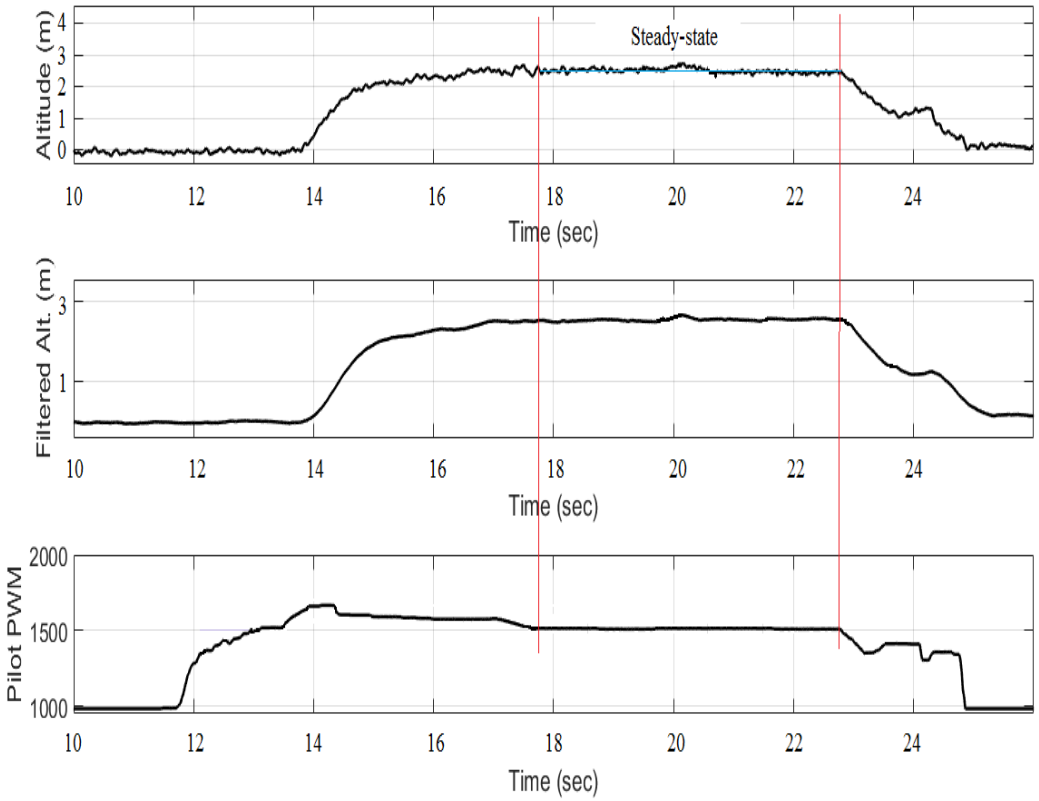

FIGURE 4.38: Steady-state hover

The procedure shown by Figure 4.38 is one where the quadrotor is lifted by manual flight control. Pilot PWM is gradually increased from about the 12 sec mark (top graph) and when thrust builds up to produce enough lift force to raise the

quadrotor, the quadrotor takes-off (just before the 14 sec mark - bottom graph). After take-off, the pilot almost immediately begins to cut the throttle input, to the point at which the quadrotor is expected to hover at constant altitude. This point corresponds to 50-percent-throttle-lever-deflection on the radio transmitter used to control the quadrotor. The quadrotor successfully hovers until the 23 sec mark when PWM is decreased to lower the quadrotor for landing.

4.5.2 Adjusted PID

Figure 4.39 shows the general response of the PID controller, for the 0.4 kg load, while Figure 4.40 shows that for 0.3 kg. Three experiments are considered for each controller, for each load mass. Knowing how steady-state is represented graphically, it is possible to more accurately analyse and evaluate the results.

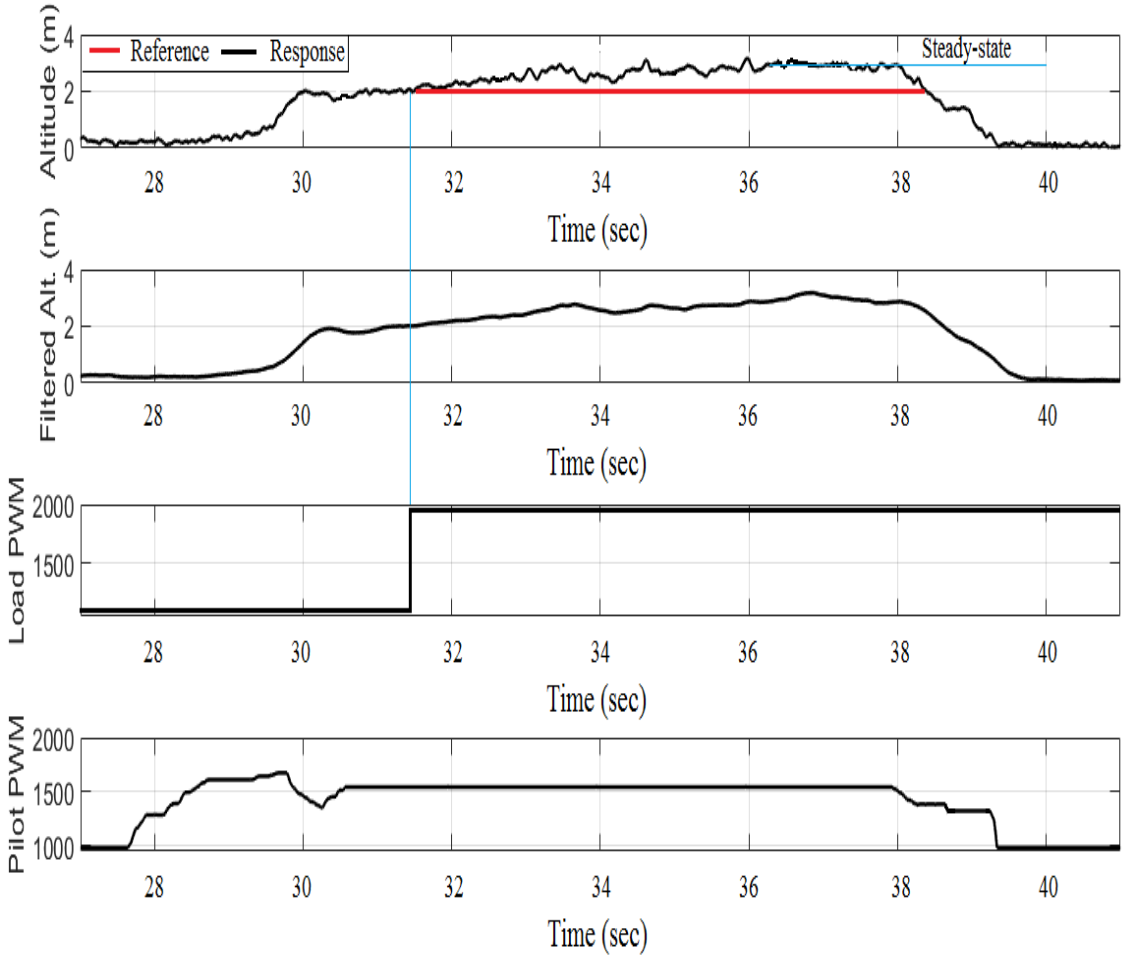

FIGURE 4.39: Final PID experiment results (0.4 kg)

As before, the procedure is to throttle up and get the quadrotor to hover. Hover is achieved at the 30 sec mark and the load is dropped off, as shown by the 'Load

PWM' plot, just before the 32 sec mark. In response, the quadrotor gains approximately 1 m in altitude over 4.5 seconds, before achieving hover state (steady-state) with 1 m error from the initial altitude before load release. The quadrotor is left to hover for approximately 2 seconds before being brought down to land, from the 38 sec mark.

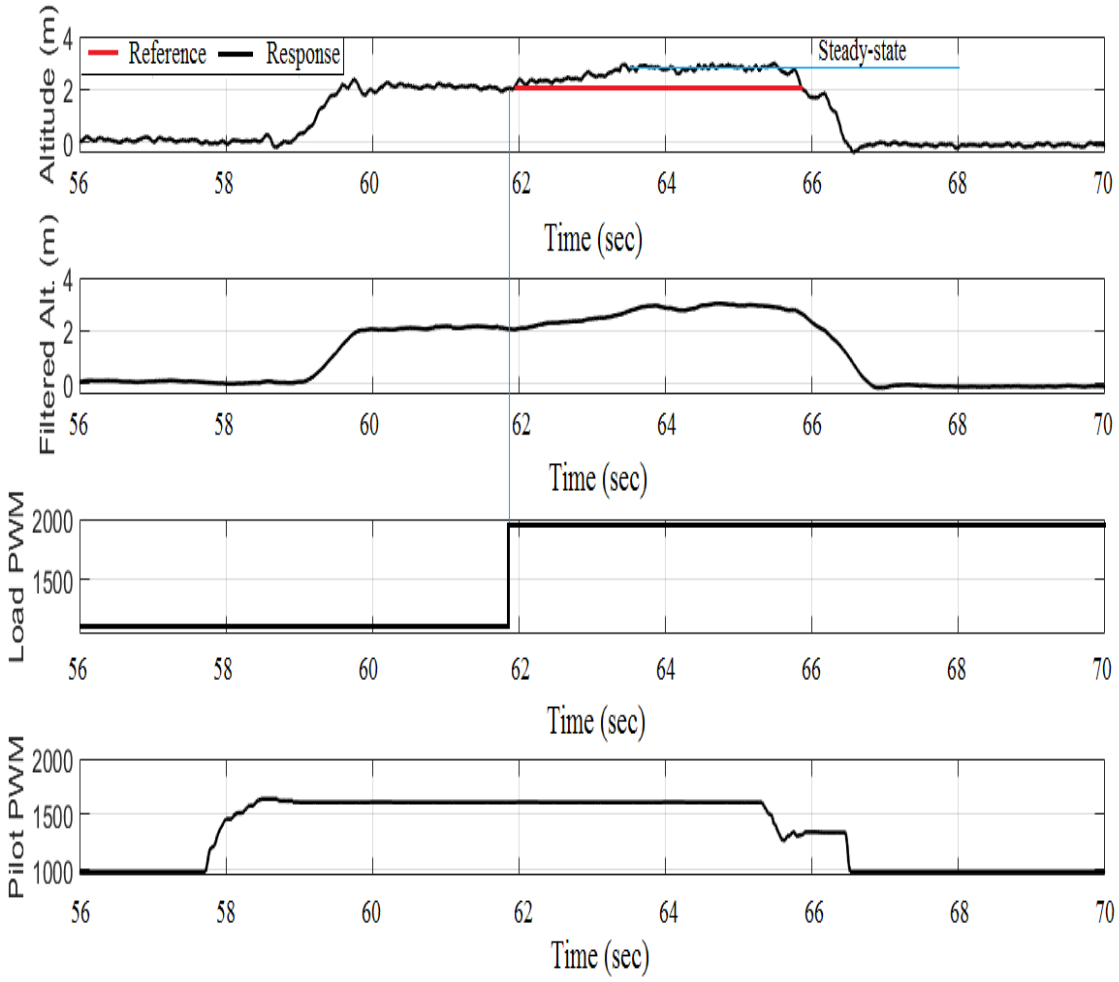

FIGURE 4.40: Final PID experiment results (0.3 kg)

In order to analyse the responses of the different controllers, some evaluation indices are required. These indices are useful in evaluating the different controllers, as explained under the next subsection.

Basis for controller evaluation

Each response is analysed to draw up conclusions. Indices of interest are steady-state error, maximum error, compensation time and overshoot. Figure 4.41, which is extracted from the PID response to the 0.3 kg load disturbance (Figure 4.40), at the point of load drop-off, is used to explain the basis for controller evaluation.

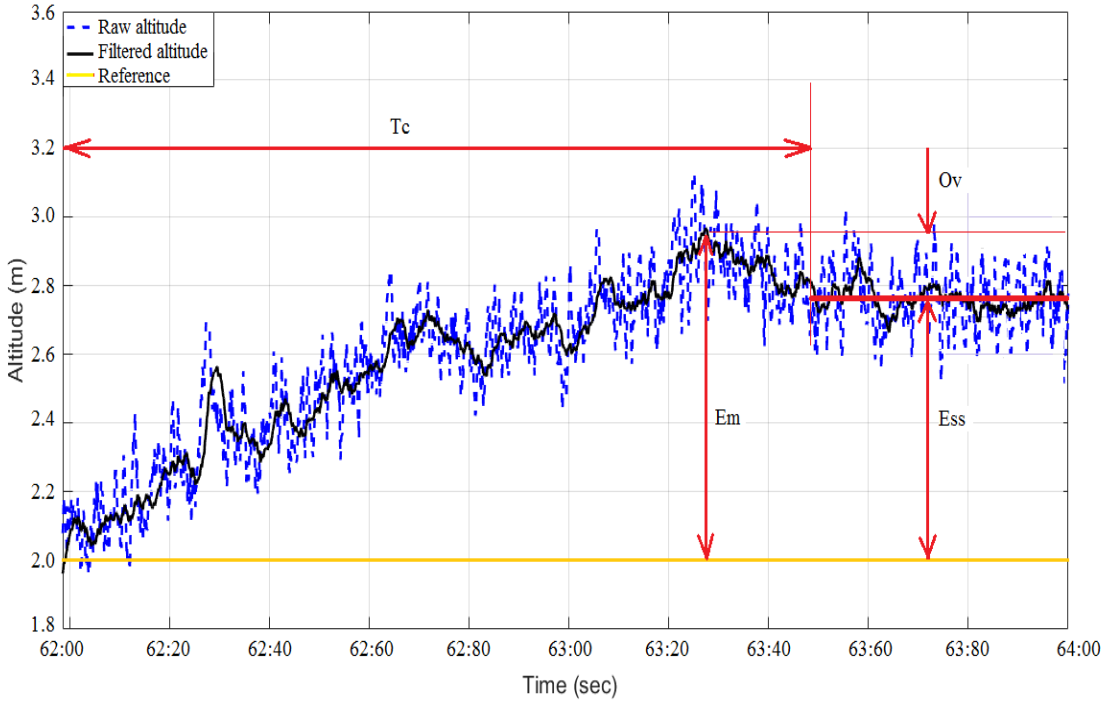

FIGURE 4.41: Basis for controller evaluation

Steady-state error is denoted E_{SS}, maximum error is E_M, compensation time is T_C and overshoot is denoted O_V. The indices are defined (for all experiments) as follows:

1. Maximum error is the largest difference between the reference altitude input and measured altitude output from the point of disturbance application until steady-state is reached in compensation for the disturbance.
2. Steady-state error is the difference between reference input and measured output at steady-state, after disturbance compensation.
3. Compensation time is the time the measured output takes to reach steady-state, from the point of application of disturbance.
4. Overshoot is the difference between steady-state measured output and the maximum measured output reached after disturbance application, as a percentage of the steady-state output.

Error-specific control indices

Some error specific control indices exist, which are useful for controller comparison (or tuning), and are typically based on empirical and also some analytical aspects. These indices include Integral Absolute Error (IAE), Integral Squared Error (ISE), and Integral Time-weighted Absolute Error (ITAE).

It is important to note that the computed total index generally depends on the particular system under investigation such that these indices are typically difficult to standardise or generalise especially for real systems. The difficulty comes from the

fact that with real systems, random disturbances act and affect the plant during experiments. These random influences affect the system output response thus altering the cumulative error measures in a manner that is unrepeatable. To draw more useful conclusions, it is common practice to introduce some unique indices, in this case to analyse system response to load disturbance. The indices defined under the previous subsection are therefore used.

PID observations for the adjusted system

The results from the experiments with the PID controller, for the different loads are as follows:

Stability

The PID controller, manages to maintain stability in altitude after load drop-off, as illustrated by the 'Altitude (m)' plots of Figure 4.39 and Figure 4.40. These figures show that altitude converges to steady-state without unstable oscillation or drift after load drop-off. The 0.3 kg load when compared against the 0.4 kg load, requires 50 percent less time

Steady-state error

From Figure 4.39, the steady-state error with PID control ('Altitude (m)' plot) is approximately 1.0 m for 0.4 kg disturbance load. This relatively large error is from the fact that the disturbance is of such magnitude that the PID control action (hover-tuned PID controller) is unable to completely compensate for the disturbance. Another contribution is the instability of the actual measurements fed from the altitude sensor, the ultrasound sensor, which in its form is affected by the corrections made by the quadrotor and is fairly noisy. ADMC is expected to be less sensitive to this measurement instability, as detailed in Chapter 3. As the magnitude of the load disturbance drops to 0.3 kg, PID performance does improve, with steady-state error falling by approximately 22 percent.

Overshoot

PID control has approximately 10 percent overshoot for the 0.4 kg load and 6 percent for the 0.3 kg load. Figure 4.39 and Figure 4.40 show that the measured altitude exceeds the steady-state level before settling, hence the overshoot.

Deadtime

At the point of drop-off, there is no deadtime in the response. Any delay in reaction at the point of drop-off is a result of the servo sweep, which then causes the load release when complete. What appears to be delay in response to particular input as represented by the Pilot PWM plot, is a result of thrust build, as explained under the Pilot PWM subsection of this Chapter. This applies to all plots.

The system performance indices used are summarised by Table 4.3, together with the other controllers.

4.5.3 MRAC

Flight tests with adaptive PID (MRAC) give a general trend as illustrated by Figure 4.42 and Figure 4.43.

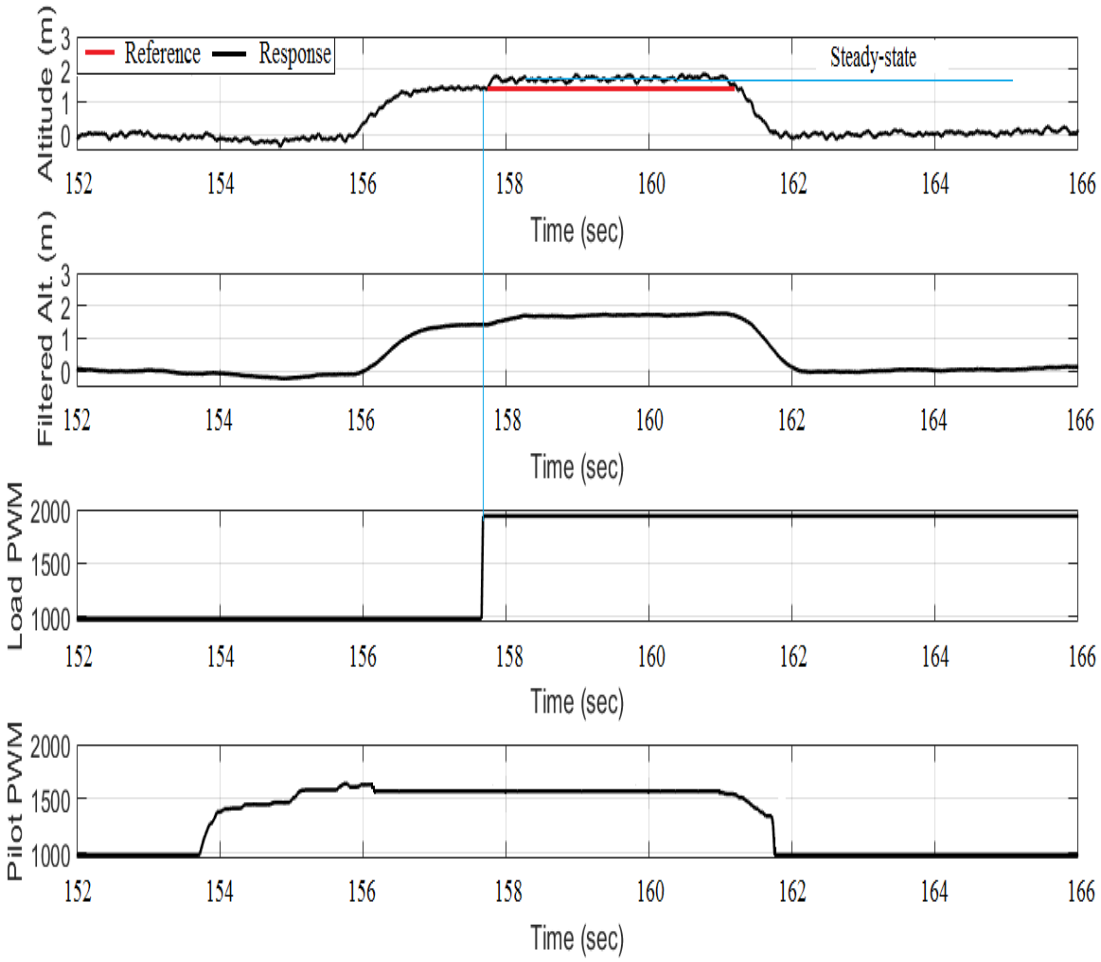

FIGURE 4.42: Final MRAC experiment results (0.4 kg)

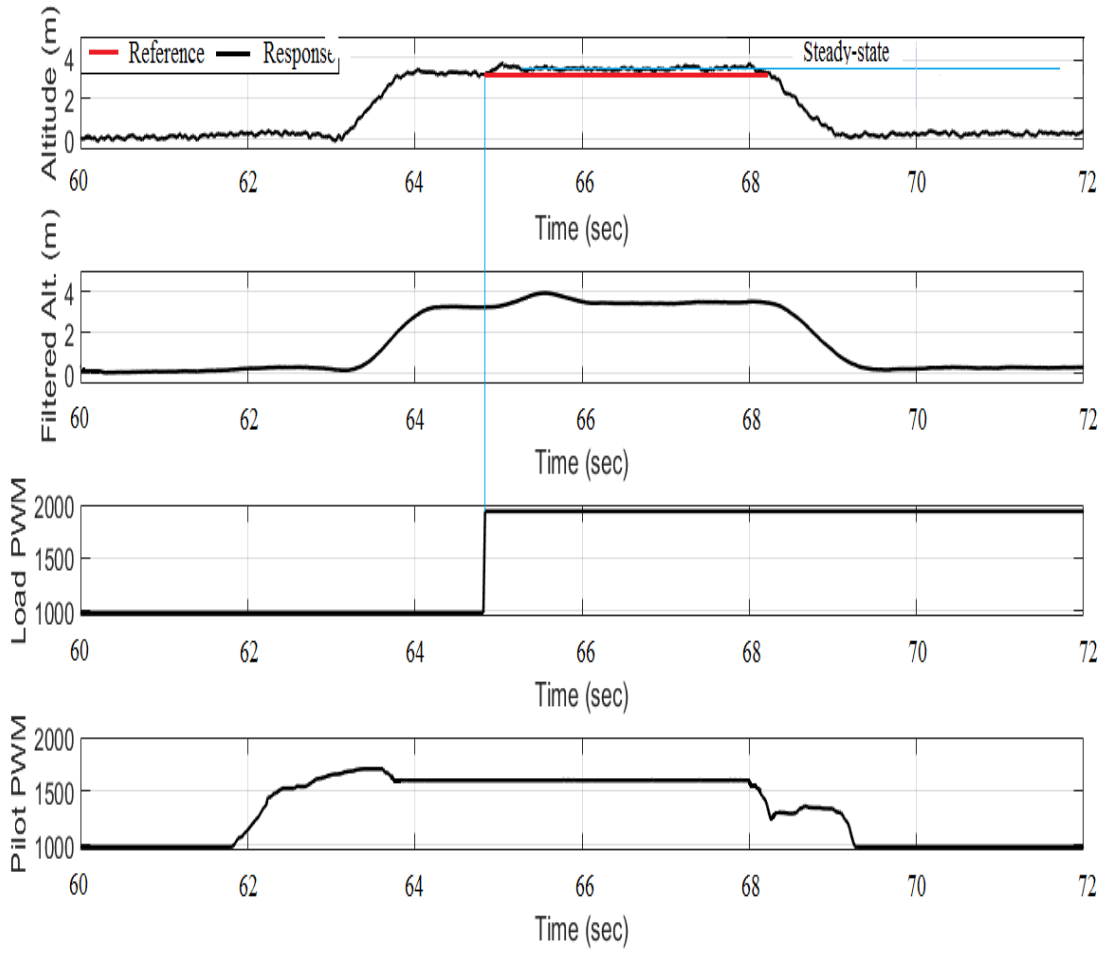

FIGURE 4.43: Final MRAC experiment results (0.3 kg)

MRAC observations

The results from the experiments with MRAC are as follows:

Stability

Adaptive PID manages to maintain stability after load release as illustrated by Figure 4.42 and Figure 4.43 where measured altitude settles at steady-state after disturbance application ('Altitude (m)' plot). Evidence of adaptation to the disturbance is illustrated by the more rapid disturbance compensation illustrated by shorter compensation time compared to PID. Decrease in disturbance magnitude increases controller performance as a matter of reduced steady-state error.

Steady-state error

The adaptive PID controller offers approximately 50 percent less steady-state error for both load disturbances as illustrated by Figure 4.42 and Figure 4.43. Adaptation thus proves useful in providing the capability to adjust to the effect of disturbance.

Overshoot

Figure 4.42 and Figure 4.43 show more pronounced adaptive PID overshoot compared to pure PID. This overshoot goes well with the much shorter compensation time related to adaptive PID.

4.5.4 Adjusted standard ADMC

Flight tests with the improved standard ADMC give a general trend as illustrated by Figure 4.44 and Figure 4.45.

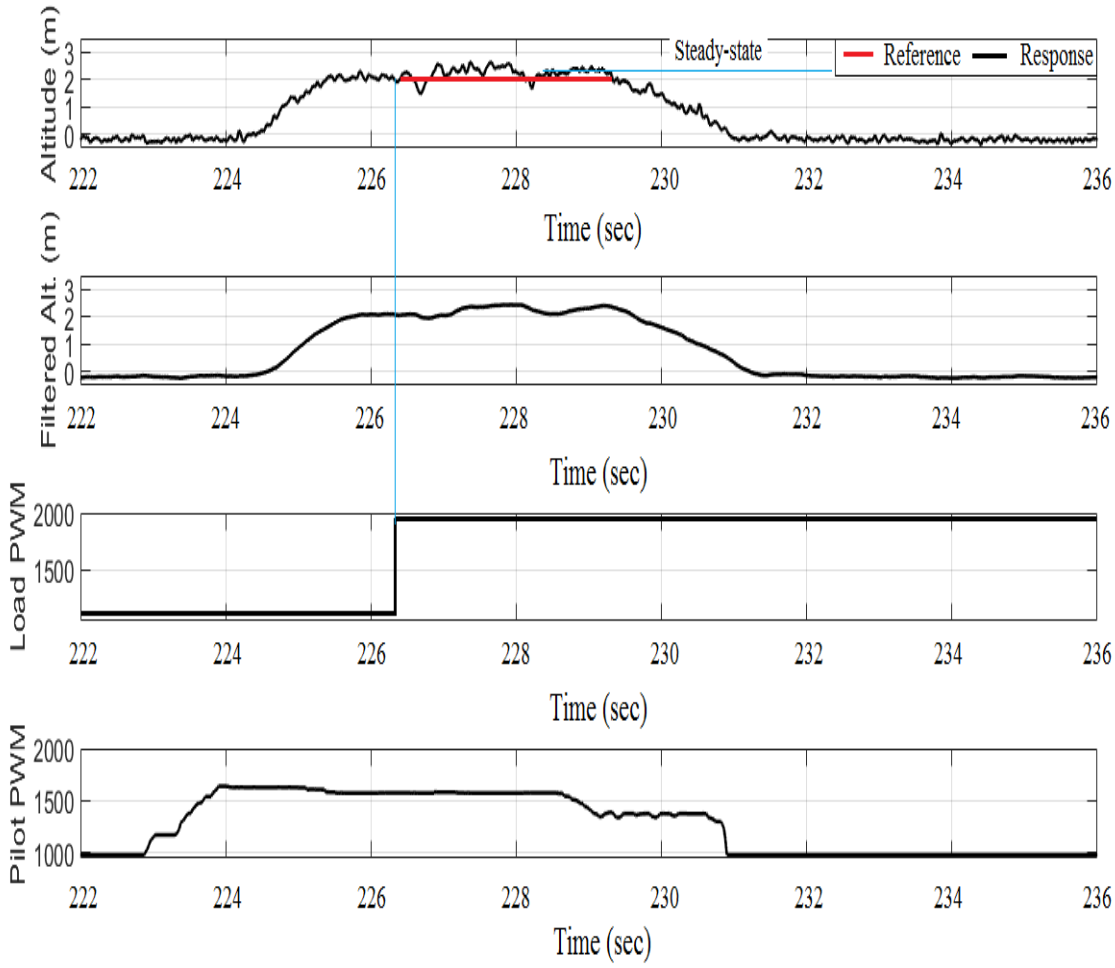

FIGURE 4.44: Final standard ADMC experiment results (0.4 kg)

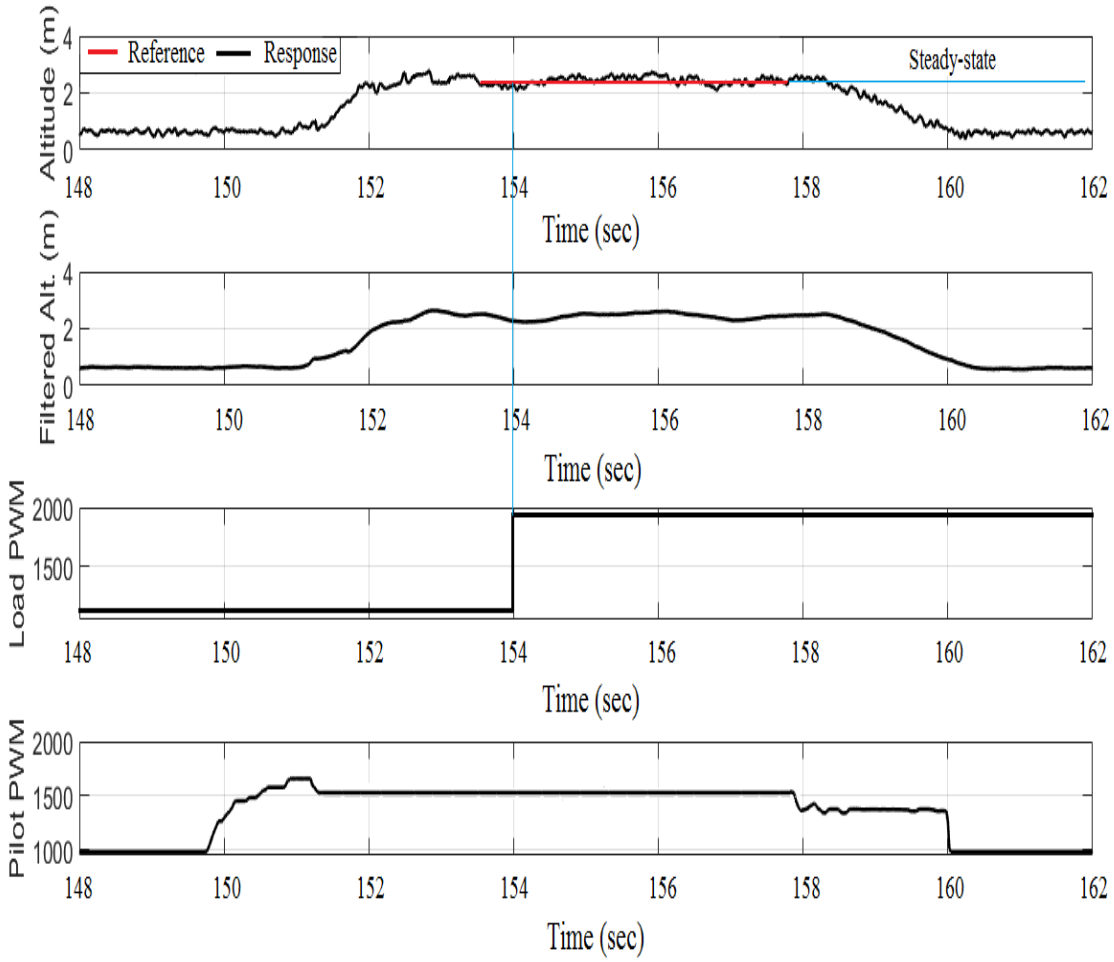

FIGURE 4.45: Final standard ADMC experiment results (0.3 kg)

Standard ADMC observations for the adjusted system

The results from the experiments with the standard ADMC controller are as follows:

Stability

Standard ADMC, like PID and adaptive PID is able to achieve stability in the quadrotor altitude after load release. Standard ADMC is observed from Figure 4.44 to oscillate in compensating for the load drop-off (226 sec - 228 sec 'Altitude (m)' plot). Together with Figure 4.45, the two demonstrate a relatively rough (better defined as less constant) steady-state output compared to adaptive PID, suggesting high aggression of the standard ADMC, which then tends to over compensate.

Steady-state error

Standard ADMC and adaptive PID have comparable steady-state error, for example 0.45 m against 0.5 m for the 0.4 kg load. If the standard ADMC is smoothed, steady-state error is expected to reduce with the smoothed ADMC as smoothing would ease out the over-compensation.

Overshoot

Standard ADMC, to this point, demonstrates highest overshoot for the three controllers discussed already. For the 0.4 kg load, Standard ADMC has approximately 25 percent overshoot (Figure 4.44, 226 sec - 228 sec 'Altitude (m)' plot), compared to 15 percent and 10 percent for adaptive PID and PID respectively. This high overshoot is explained by the aggression of the DMC controller. Overshoot reduces as the load reduces in weight, where the 0.3 kg load response has a 15 percent overshoot.

4.5.5 Adjusted smoothed ADMC

Flight tests with the improved smoothed ADMC give the general responses illustrated by Figure 4.46 and Figure 4.47.

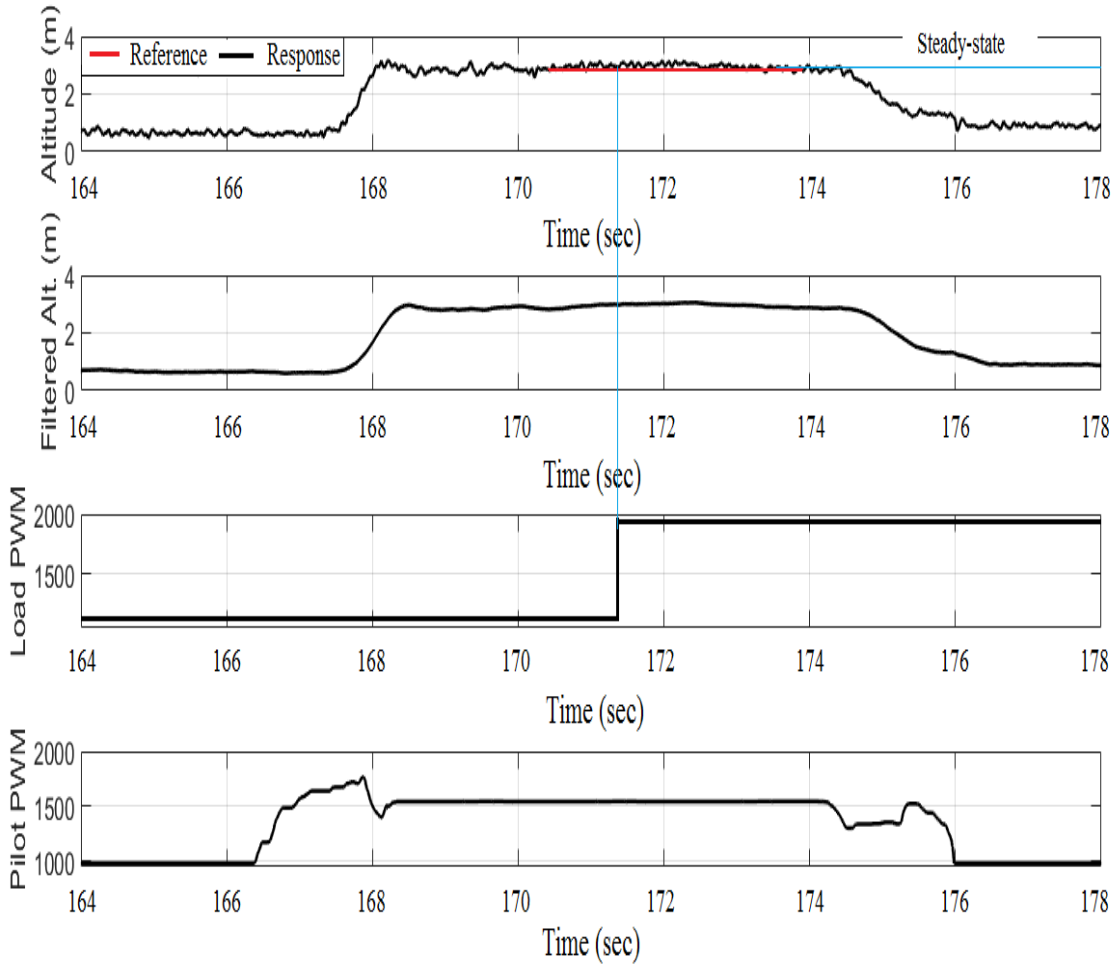

FIGURE 4.46: Final smoothed ADMC experiment results (0.4 kg)

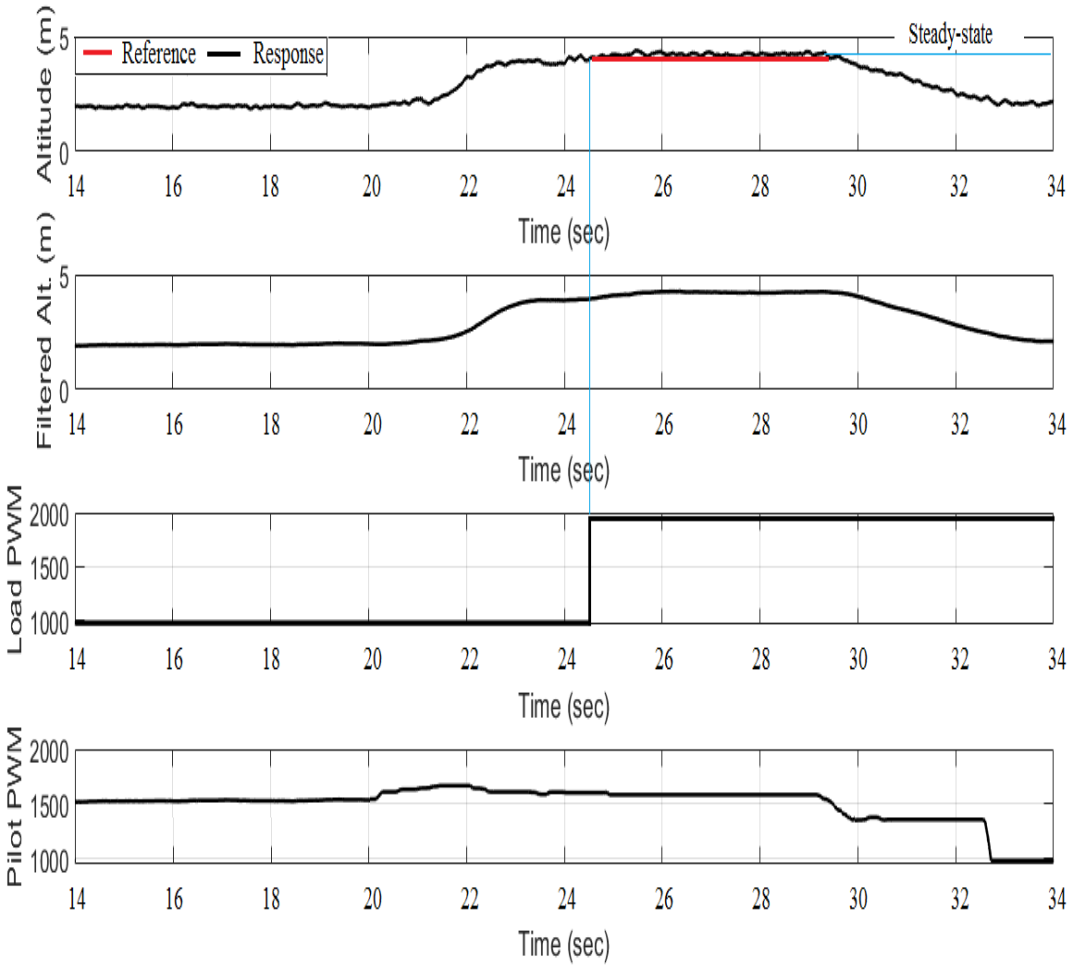

FIGURE 4.47: Final smoothed ADMC experiment results (0.3 kg)

Smoothed ADMC observations for the adjusted system

The results from the experiments with the standard ADMC controller are as follows:

Stability

Smoothed ADMC manages to maintain altitude stability after load drop-off, as the case with standard ADMC. Smoothed ADMC is able to more closely return to the initial drop-off altitude where for example 0.4 kg load smoothed ADMC offers approximately 0 m steady-state error against 0.45 m, 0.5 m and 1 m for standard ADMC, adaptive PID and PID respectively.

Steady-state error

Smoothed ADMC, as stated above, offers the least steady-state error. It can be concluded from this fact, that smoothed ADMC is capable of highest precision input command tracking, given the controllers involved.

Overshoot

No overshoot is observed for the smoothed ADMC case. The smoothing function is capable of ensuring rapid but smooth correction for the load drop-off.

4.5.6 Overall analysis

Table 4.3 is useful in carrying out an overall analysis of the results presented for the experiments above, for the load drop-off disturbance experiment with the Axe quadrotor.

TABLE 4.3: Final evaluation

Index	**Maximum error (m)**	**Steady-state error (m)**	**Compensation time (sec)**	**Percentage overshoot**
PID (0.4 kg)	1.35	1.0	4.5	10
PID (0.3 kg)	0.95	0.78	1.5	6
MRAC (0.4 kg)	0.8	0.5	< 0.5	15
MRAC (0.3 kg)	0.6	0.3	< 0.5	15
Standard ADMC (0.4 kg)	0.8	0.45	2	25
Standard ADMC (0.3 kg)	0.4	<0.1	2	15
Smooth ADMC (0.4 kg)	0.2	<0.1	< 0.5	0
Smooth ADMC (0.3 kg)	<0.1	<0.1	< 0.5	0

To fully analyse all the work done, it is necessary to present data from all three experiments carried out for each controller, for each load. Figure 4.48 illustrates a box plot for the steady-state errors from the three experiments considered for each controller.

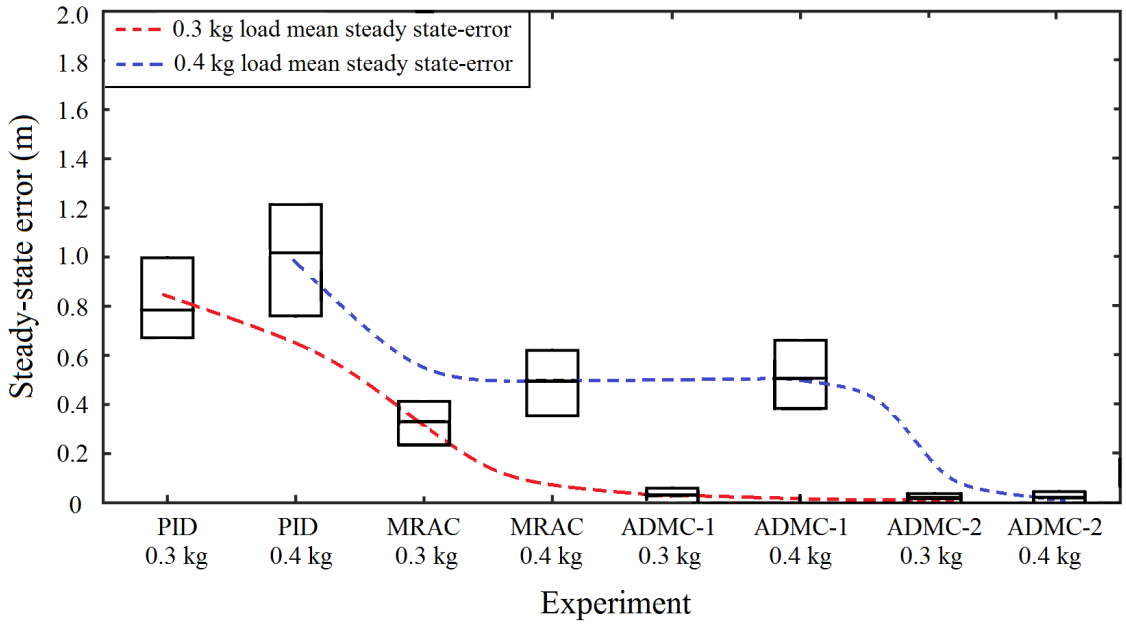

FIGURE 4.48: Steady-state error box plot

For each box, the horizontal lines (top, middle and bottom) indicate the values for steady-state error recorded for each of the three experiments. ADMC-1 represents standard ADMC while ADMC-2 represents smoothed ADMC. The red and blue dotted lines, plot the mean steady-state error for the three experiments for each controller, for the 0.3 kg and the 0.4 kg load respectively.

General analysis

Simulation experiments and physical experiments do not produce exactly identical results. In simulation, all controllers perform better than the experimental counterpart. It can be confirmed that the real system operating in the real world is affected not only by unforeseen eternal influences, but also the quality of the modules and systems utilised. This includes sensors, computers and software.

Controller specific analysis

The PID controller performs to high quality for the intended purpose, without load disturbance. With the introduction of varying loads, especially loads of significant weight such as demonstrated in this work, PID performance deteriorates. As the magnitude of the disturbance falls, PID control does improve. Modifying the PID controller by means of making the controller adaptive must improve PID performance for the varying load case. Tests with the adaptive PID controller do reveal improved PID performance. Adaptation allows better compensation for the load drop-off, better in the sense of reduced steady-state error, more rapid compensation and reduced maximum error on application of load disturbance. The standard and smoothed ADMC also reveal improved control. Compared against adaptive PID, the ADMC experiments demonstrate two sides. ADMC results in reduced steady state error although, for standard ADMC case, while steady-state error is less, disturbance compensation is quicker for adaptive PID, if the mild oscillations

of standard ADMC are to be considered. The DMC algorithm in ADMC, which is of optimal control, seeks to track reference input as closely as possible. Without smoothing, this then results in over-compensation in some cases. From Figure 4.48, it is observed that the effects of the difference between the 0.3 kg and 0.4 kg load disturbance is most pronounced for the standard ADMC controller, and this is due to the over-compensation which makes it more difficult for the quadrotor to settle back at the precise initial altitude (as from Figure 4.44, 226 sec - 228 sec). When a smoothing function is implemented, the result is then of minimal error and rapid compensation and this is displayed by smoothed ADMC. Overall, for systems operating in environments where disturbance varies significant in magnitude, adaptive control is recommended.

4.5.7 Case study - computational cost

To understand exactly how much more expensive ADMC is computationally, an exercise is carried out to evaluate the time execution of the PID and ADMC. The idea is to measure the amount of time required by each controller to produce control action. While this gives some idea of the computational requirement for each controller, the time measurement is actually affected by initialisation time, which is observed to vary from one simulation to another even for one controller.

To carry out this exercise, a 'timestamp' is put to use. This timestamp is able to analyse execution performance in real-time and highlight the time a particular signal becomes live, from the start of the run. The implementation of the Timestamp block in this case is as shown by Figure 4.49, using a subtraction technique.

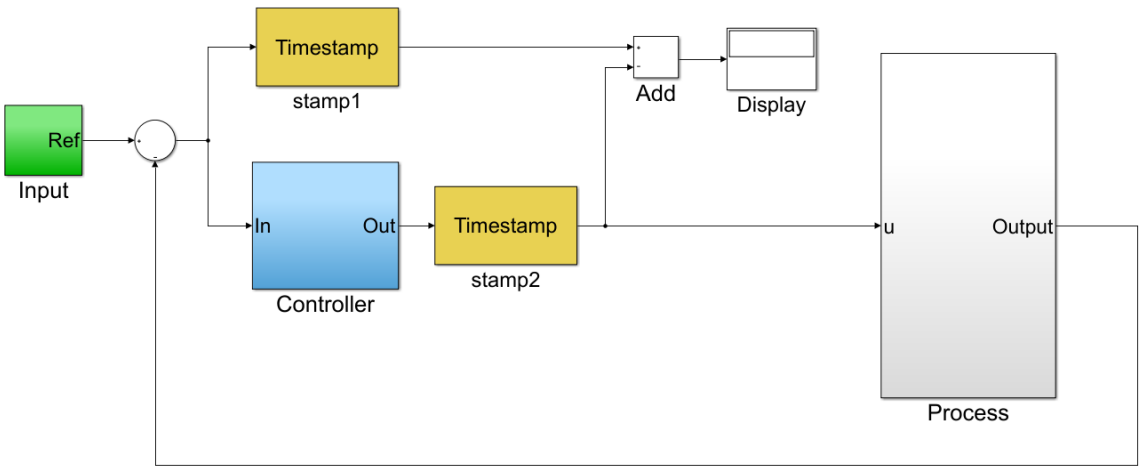

FIGURE 4.49: Implementation of the Timestamp block

Four simulations are run for each controller and the average execution time for each case is presented in Table 4.4. The simulations are ran in external mode on a Raspberry Pi computer.

TABLE 4.4: Controller execution time

Controller	Time (microseconds)
ADMC (Z)	420
PID	110

This exercise shows that, in reality, the difference in computational cost between PID and ADMC is comparable, but yes, it is significant. Measures were taken though, to minimise computational cost, which include:

- Removal of scope blocks.
- Removal of unnecessary out-port blocks.
- Elimination of redundant calculations.

4.6 Summary

It can be concluded that implementation of ADMC achieves high quality control for the load pick-up and drop-off quadrotor, coming from both simulation and experiments. This quality is with respect to steady-state error, maximum error and compensation time. Compared even against adaptive PID, ADMC performs slightly better. ADMC is seen to genuinely provide faster and more accurate disturbance compensation and adaptation to the disturbance, ensuring precise tracking of the reference input command. This comes from both simulation and experimentation. This however comes at the cost of increased complexity (and computation) such that, the performance of the predictive controller will indeed depend on hardware. While experiments introduce certain difficulties, experiments expose aspects otherwise sometimes not fully exposed in simulation. With the computers available today, the complexity of ADMC can however be handled just as efficiently as the simple PID. It should also be mentioned that the performance of a system is affected by the quality of system modules used such as sensors. ADMC is, overall, a topic worth researching into for improved control of quadrotors, especially on an experimental basis.

Chapter 5

Conclusion

5.1 Conclusion

This work shows that ADMC is capable of improved quadrotor control in trajectory tracking and disturbance rejection in the case of a delivery quadrotor. The cost of ADMC was, in this work justified, based on the ability of ADMC to produce disturbance rejection and adaptation to the effects caused by disturbance (loading or unloading), ultimately tracking the required reference command more accurately than the PID counter-part, and slightly more that adaptive PID too.

First of all, it can be concluded, based on findings from literature review, that the application of quadrotors to pick and place operations is beneficial, for example in delivery of medical supplies in emergency situations or delivery of goods, especially to remote destinations. This requires viable operational protocols to ensure safety of people and property.

Further on, experimental modelling of the quadrotor based on system identification is capable of producing high accuracy system models. It is however important to verify the accuracy of identified models against the analytical models. This is because system identification may be affected negatively by outliers and sensor bias etc. Chapters 4 and 5 show that ADMC provides adaptation to changing dynamics (simulation and experiments), in this case caused by load pick-up, drop-off or both. The DMC part of the controller ensures high reference tracking accuracy based on optimal control. Implementation of ADMC demands caution. First, the controller must be tuned to satisfactory standards regardless of the fact that the tuning process is mostly intuitive. The end controller product will perform according to the computational capacity of the computer used for implementation. As for noise sensitivity, ADMC, because it is non-parametric, is less sensitive to noise compared to the standard PID controller. It must be mentioned that the simulation-tuned ADMC controller had to be adjusted significantly, to achieve the expected control quality with the practical quadrotor system. ADMC aggression is an aspect of interest, and is tuned by adjusting some aggression factor, which in turn regulates how aggressive the ADMC controller is in reference tracking. While high values of the aggression factor may result in overcompensation, lower values may compromise the control quality negatively by slower response time. This work explored a strategy to employ a smoothing algorithm while keeping the aggression factor high. This was observed to improve ADMC.

Overall, the exploitation of ADMC theoretically is not entirely adequate because in the real world, ADMC will depend on the system involved, the different components of the system and external conditions. In this work, the simulation controller had to be changed significantly on the experimental stage, even though simulations revealed desired quality in control. Finally, the complexity of ADMC is justified based on the improved control quality in reference command tracking and disturbance rejection.

5.1.1 Research summary

In this work, Adaptive Dynamic Matrix Control (ADMC) is investigated for the quadrotor pick-and-place task. The overall goal is to test ADMC experimentally. Steps taken in tuning ADMC are laid out, together with some investigations which showcase the effects of the adjustable parameters of the controller. Simulations are carried out, gradually increasing complexity and closeness to the reality of the physical quadrotor. Ultimately, the simulations compare ADMC against adaptive PID in the form of Model Reference Adaptive Control (MRAC) and also Proportional Integral Derivative (PID) control, for the pick-and-place delivery quadrotor. Results show that ADMC and MRAC offer higher control quality. The ADMC controller is deployed on a physical quadrotor and tests carried out in comparison to PID and adaptive PID. Some issues to look out for when implementing ADMC are investigated. The simulation tuned ADMC is initially observed to be excessively aggressive hence the proposition of a smoothing algorithm. The final experiments reveal that smoothed ADMC is capable of improved quadrotor control for the varying load case. Adaptive PID, which is capable of self-adjusting for different control points, demonstrates improved control as well, from standard PID.

5.2 Future work

Based on the findings in this work, it is evident that research into application of ADMC to aerial robots such as quadrotors is worthwhile. More can be done in investigating various ADMC settings experimentally. This however depends on the system's capability to be operated safely. A unique test bench will go a long way towards this future work. Ideally, this should include complete pick and place experiments, with obstacle avoidance. Future work will also extend to investigate effects of different manoeuvres, for example, effects of yawing on altitude, for the different controllers studied in this work.

REFERENCES

[1] DHL Research, "Unmanned Aerial Vehicle in Logistics." DHL Trend Research, DHL Customer Solutions and Innovation, Bonn, 2014.

[2] M. Schreier, "Modeling and adaptive control of a quadrotor," *in 2012 IEEE International Conference on Mechatronics and Automation*, Chengdu, 2012, pp. 383–390.

[3] I. D. Landau, R. Lozano, M. M'Saad, and A. Karimi, "Adaptive Control," Springer, London, 2011.

[4] Y. Osa, T. Mabuchi, and S. Uchikado, "Synthesis of discrete time adaptive flight control system using nonlinear model matching," *in IEEE International Symposium on Industrial Electronics*, 2001. Proceedings. ISIE 2001, Pusan, 2001, vol. 1, pp. 58–63 vol.1.

[5] K. Astrom and B. Wittenmark, "Adaptive Control: Second Edition," Dover Publications INC, Mineola, New York, 2008.

[6] E. Lavretsky, "Adaptive control: Introduction, overview, and applications," *in Robust and Adaptive Control Workshop*, California, 2008.

[7] J. M. M. Sâanchez and J. Rodellar, Adaptive Predictive Control: From the Concepts to Plant Optimization, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1995.

[8] S. Kim, S. Choi, and H. J. Kim, "Aerial manipulation using a quadrotor with a two DOF robotic arm," *in 2013 IEEE International Conference on Intelligent Robots and Systems*, Tokyo, 2013, pp. 4990–4995.

[9] G. Garimella and M. Kobilarov, "Towards model-predictive control for aerial pick-and-place," *in 2015 IEEE International Conference on Robotics and Automation*, 2015, pp. 4692–4697.

[10] A. Ghaffar and T. Richardson, "Model Reference Adaptive Control and LQR Control for Quadrotor with Parametric Uncertainties," *World Academy of Science, Engineering and Technology, International Journal of Mechanical, Aerospace, Industrial, Mechatronic and Manufacturing Engineering*, vol. 9, 2015, pp. 244–250.

[11] P. E. I. Pounds, D. R. Bersak, and A. M. Dollar, "Grasping from the air: Hovering capture and load stability," *in 2011 IEEE International Conference on Robotics and Automation*, Shangai, 2011, pp. 2491–2498.

[12] V. Ghadiok, J. Goldin, and W. Ren, "Autonomous indoor aerial gripping using a quadrotor," *in 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Francisco, CA, 2011, pp. 4645–4651.

[13] G. Heredia, A. E. Jimenez-Cano, I. Sanchez, D. Llorente, V. Vega, J. Braga, J. A. Acosta, and A. Ollero, "Control of a multirotor outdoor aerial manipulator," *in 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Chicago, IL, 2014, pp. 3417–3422.

[14] D. Mellinger, M. Shomin, N. Michael, and V. Kumar, "Cooperative Grasping and Transport Using Multiple Quadrotors," *in Distributed Autonomous Robotic Systems*, A. Martinoli, F. Mondada, N. Correll, G. Mermoud, M. Egerstedt, M. A. Hsieh, L. E. Parker, and K. Støy, Eds. Springer Berlin Heidelberg, 2013, pp. 545–558.

[15] G. Gioioso, A. Franchi, G. Salvietti, S. Scheggi, and D. Prattichizzo, "The flying hand: A formation of UAVs for cooperative aerial tele-manipulation," *in 2014 IEEE International Conference on Robotics and Automation*, Hong Kong, 2014, pp. 4335–4341.

[16] D. Mellinger, N. Michael, and V. Kumar, "Trajectory Generation and Control for Precise Aggressive Maneuvers with Quadrotors," in Experimental Robotics, O. Khatib, V. Kumar, and G. Sukhatme, Eds. Berlin: Springer Berlin Heidelberg, 2014, pp. 361–373.

[17] Emre Can Suicmez, "Trajectory tracking of a quadrotor unmanned aerial vehicle (UAV) via attitude and position control," MSc Thesis, Middle East Technical University, Ankara, Turkey, 2014.

[18] S. Bouabdallah and R. Siegwart, "Backstepping and Sliding-mode Techniques Applied to an Indoor Micro Quadrotor," *in Proceedings 2005 IEEE International Conference on Robotics and Automation*, Barcelona, 2005, pp. 2247–2252.

[19] T. Madani and A. Benallegue, "Backstepping Control for a Quadrotor Helicopter," *in 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, 2006, pp. 3255–3260.

[20] Z. C. Hou, X. Gong, Y. Bai, Y. T. Tian, and Q. Sun, "Robust Trajectory Tracking Control of Autonomous Quad-Rotor UAV Based on Double Loop Frame," Key Engineering Materials, vol. 467–469, Feb. 2011, pp. 1421–1426.

[21] S. Lee, S. H. Kang, and Y. Kim, "Trajectory tracking control of quadrotor UAV," *in 2011 11th International Conference on Control, Automation and Systems*, Gyeonggi-do, 2011, pp. 281–285.

[22] A. Benallegue, A. Mokhtari, and L. Fridman, "Feedback linearization and high order sliding mode observer for a quadrotor UAV," *in 2006 International Workshop on Variable Structure Systems*, 2006, Alghero, Sardinia, 2006, pp. 365–372.

[23] Z. Zou, "Quadrotor Trajecotry Tracking Control: A PD Control Algorithm," *in 2010 3rd International Conference on Computer and Electrical Engineering*, Beijing, 2010.

[24] C. Liu, S. J. Tang, S. Y. Yang, and J. Guo, "Fuzzy Sliding-Mode Control for Quad-Rotor Trajectory Tracking," Applied Mechanics and Materials, vol. 278–280, pp. 1593–1600, Jan. 2013.

[25] C. Wang, B. Song, P. Huang, and C. Tang, "Trajectory Tracking Control for Quadrotor Robot Subject to Payload Variation and Wind Gust Disturbance," *Journal of Intelligent Robotic Systems*, vol. 83, no. 2, Jan. 2016, pp. 315–333.

[26] P. D. Monte and B. Lohmann, "Position trajectory tracking of a quadrotor helicopter based on L1 adaptive control," *in 2013 European Control Conference*, Zurich, 2013, pp. 3346–3353.

[27] P. Ioannou, Robust adaptive control. USA: Prentice Hall, 1996.

[28] G. Tao and P. V. Kokotovic, "Continuous-time Adaptive Control of Systems with Unknown Backlash," *in 1993 American Control Conference*, USA, 1993, pp. 1344–1348.

[29] X. Chen and Y. Nagaya, "Adaptive control for piezo-actuated stage based on continuous-time approach," *in 2012 IEEE International Conference on Mechatronics and Automation*, Japan, 2012, pp. 2059–2064.

[30] R. Chi and J. Han, "A novel periodicity-based adaptive control discrete-time nonlinear systems," *in 2015 27th Chinese Control and Decision Conference*, China, 2015, pp. 421–424.

[31] J. Li, H. Ma, C. Yang, and M. Fu, "Discrete-time adaptive control of robot manipulator with payload uncertainties," *in 2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems*, China, 2015, pp. 1971–1976.

[32] C. Nicol, C. J. B. Macnab, and A. Ramirez-Serrano, "Robust neural network control of a quadrotor helicopter," *in 2008 Canadian Conference on Electrical and Computer Engineering*, Niagara Falls, 2008, pp. 1233–1238.

[33] S. Li, B. Li, and Q. Geng, "Adaptive sliding mode control for quadrotor helicopters," *in 2014 33rd Chinese Control Conference*, Nanjing, 2014, pp. 71–76.

[34] H. Zhen, X. Qi, and H. Dong, "An Adaptive Block Backstepping Controller for Attitude Stabilization of a Quadrotor Helicopter," *WSEAS Transactions on Systems and Control*, April 2013, Vol. 8 Issue 2, pp. 46.

[35] Z. Fang and W. Gao, "Adaptive integral backstepping control of a Micro Quadrotor," *in 2011 2nd International Conference on Intelligent Control and Information Processing*, vol. 2, Harbin, 2011, pp. 910–915.

[36] H. Boudjedir, "Dual Neural Network for Adaptive Sliding Mode Control of Quadrotor Helicopter Stabilization," *in 2012 International Journal of Information Sciences and Techniques*, vol. 2, no. 4, Jul. 2012, pp. 1–14.

[37] R. Hedjar, "Robust one-step-ahead model predictive control of VTOL-UAVs," *in 2015 27th Chinese Control and Decision Conference*, China, 2015, pp. 3053–3058.

[38] K. Alexis, G. Nikolakopoulos, and A. Tzes, "On Trajectory Tracking Model Predictive Control of an Unmanned Quadrotor Helicopter Subject to Aerodynamic Disturbances," *Asian Journal of Control*, vol. 16, no. 1, January 2014, pp. 209–224.

[39] K. Kunz, S. M. Huck, and T. H. Summers, "Fast Model Predictive Control of miniature helicopters," *in 2013 European Control Conference, Zurich*, 2013, pp. 1377 – 1382.

[40] K. Alexis, G. Nikolakopoulos, and A. Tzes, "Experimental model predictive attitude tracking control of a quadrotor helicopter subject to wind-gusts," *in 2010 18th Mediterranean Conference on Control Automation*, Morocco, 2010, pp. 1461–1466.

[41] M. Abdolhosseini, Y. M. Zhang, and C. A. Rabbath, "An Efficient Model Predictive Control Scheme for an Unmanned Quadrotor Helicopter," *Journal of Intelligent Robotic Systems*, vol. 70, no. 1–4, Aug. 2012, pp. 27–38.

[42] B. A. Kim, S. H. Lee, Y. O. Lee, and C. C. Chung, "Comparative study of approximate, proximate, and fast model predictive control with applications to autonomous vehicles," *in 2012 12th International Conference on Control, Automation and Systems*, JeJu Island, 2012, pp. 479–484.

[43] P. Bouffard, A. Aswani, and C. Tomlin, "Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results," *in 2012 IEEE International Conference on Robotics and Automation*, Saint Paul, 2012, pp. 279 – 284.

[44] J. M. Aymes, "Implementing Adaptive Predictive Control with the TMS320C50 DSP," Texas Instruments Publications, France, 1997.

[45] G. Chowdhary, M. Mühlegg, J. P. How, and F. Holzapfel, "A concurrent learning adaptive-optimal control architecture for nonlinear systems," *in 2013 52nd IEEE Conference on Decision and Control*, Firenze, 2013, pp. 868–873.

[46] A. Rahideh, H. Shaheed, and H. Huijberts, "Stable Adaptive Model Predictive Control for Nonlinear Systems," *in 2008 American Control Conference*, Washington, 2008, pp. 1673–1678.

[47] C. Balas, "Modelling and Linear Control of a Quadrotor," MSc Thesis, Cranfield University, Cranfield, 2007.

[48] S. Bouabdallah, "Design and control of quadrotors with application to autonomous flying," MSc Thesis, Ecole Polytechnique Federale de Lausanne, Algeria, 2007.

[49] S. F. Ahmed, K. Kushsairy, M. I. A. Bakar, D. Hazry, and M. K. Joyo, "Attitude stabilization of Quad-rotor (UAV) system using Fuzzy PID controller (an experimental test)," *in 2015 2nd International Conference on Computing Technology and Information Management*, Johor, 2015, pp. 99–104.

[50] H. Abaunza, P. Castillo, R. Lozano, and A. Victorino, "Quadrotor aerial manipulator based on dual quaternions," *in 2016 International Conference on Unmanned Aircraft Systems*, France, 2016, pp. 152–161.

[51] G. Jithu and P. R. Jayasree, "Quadcopter modelling and control," International Conference on Electrical, Electronics and Optimization Techniques, Chennai, 2016.

[52] D. Xu, L. Wang, G. Li, and L. Guo, "Modeling and Trajectory Tracking Control of a Quad-rotor UAV," *in 2012 2nd International Conference on Computer Application and System Modeling*, Paris, 2012.

[53] Y. Yu and X. Ding, "A Quadrotor Test Bench for Six Degree of Freedom Flight," *Journal of Intelligent Robotic Systems*, vol. 68, no. 3–4, May 2012, pp. 323–338.

[54] G. M. Hoffmann, H. Huang, S. L. Waslander, and C. J. Tomlin, "Quadrotor helicopter flight dynamics and control: Theory and experiment," *in Proceedings of the 2007 AIAA Guidance, Navigation, and Control Conference*, USA, 2007, vol. 2, pp. 4.

[55] S. Lupashin, M. Hehn, M. W. Mueller, A. P. Schoellig, M. Sherback, and R. D'Andrea, "A platform for aerial robotics research and demonstration: The Flying Machine Arena," Mechatronics, vol. 24, no. 1, Feb. 2014, pp. 41–54.

[56] W. Dong, G.-Y. Gu, X. Zhu, and H. Ding, "Development of a Quadrotor Test Bed — Modelling, Parameter Identification, Controller Design and Trajectory Generation," *in 2015 International Journal of Advanced Robotic Systems*, 2015, pp. 1.

[57] A. Ji and K. Turkoglu, "Development of a Low-Cost Experimental Quadcopter Testbed Using an Arduino Controller and Software," Sage, USA, Aug. 2015.

[58] I. Stanculeanu and T. Borangiu, "Quadrotor black-box system identification," World Academy of Science, Engineering and Technology, vol. 5, 2011, pp. 276–279.

[59] V. Khorani, N. Ajilforoushan, and A. M. Shahri, "An IMU-based system identification technique for quadrotors," *in 2013 3rd Joint Conference of AI Robotics and 5th RoboCup Iran Open International Symposium*, Tehran, 2013, pp. 1–6.

[60] J. Yang, Z. Cai, Q. Lin, D. Zhang, and Y. Wang, "System identification of quadrotor UAV based on genetic algorithm," *in 2014 IEEE Chinese Guidance, Navigation and Control Conference*, Yantai, 2014, pp. 2336–2340.

[61] A. Ruesh, "Dynamics Identification and Validation, and Position Control for a Quadrotor," Thesis, Swiss Federal Institute of Technology, Zurich, 2010.

[62] K. Befus, "Localization and System Identification of a Quadcopter UAV," MSc Thesis, Western Michigan University, Michigan, 2014.

[63] J. Müller and W. Burgard, "Efficient probabilistic localization for autonomous indoor airships using sonar, air flow, and IMU sensors," Advanced Robotics, June 2013 vol. 27, no. 9, pp. 711–724.

[64] L. Jose, G. Sanahuja, R. Lozano, S. Salazar, R. Garcia-Hernandez, and J. A., "Indoor Localization of a Quadrotor Based on WSN: A Real-Time Application," *in 2013 International Journal of Advanced Robotic Systems*, 2013, pp. 1.

[65] R. Upadhyay and R. Singla, "Application of adaptive control in a process control," *in 2010 2nd International Conference on Education Technology and Computers*, Shangai, 2010, vol. 2, pp. 323-327.

[66] Z. He and L. Zhao, "A Simple Attitude Control of Quadrotor Helicopter Based on Ziegler-Nichols Rules for Tuning PD Parameters," *in The Scientific World Journal*, December 2014 vol. 2014, pp. e280180.

[67] H. Tnunay, M. Q. Abdurrohman, Y. Nugroho, and Y. Yamamoto, "Auto-tuning quadcopter using Loop Shaping," *in International Conference on Computer, Control, Informatics and Its Applications (IC3INA)*, Indonesia, 2013.

[68] Hossein Bolandi, Mohammad Rezaei, Reza Mohsenipour, Hossein Nemati and Seed Smailzadeh, "Attitude Control of a Quadrotor with Optimized PID Controller," *in Intelligent Control and Automation*, vol. 4, Aug 2013, pp. 335-342.

[69] Y. Li, K. H. Ang, G. Chong, W. Feng, K. C. Tan and H. Kashiwagi, "CAutoCSD-Evolutionary Search and Optimisation Enabled Computer Automated Control System Design," *in International Journal of Automatic and Computer Systems*, Vol. 1, No. 1, 2004. pp. 76-88.

[70] I. M. Y. Mareels, C. R. Johnson, R. L. Kosut, and M. A. Poubelle, "How Exciting Can a Signal Really Be?" *System Control Lett.*, vol. 8, no. 3, North-Holland, Jan. 1987 pp. 197–204.

[71] R. Kosut, B. Anderson, and I. Mareels, "Stability theory for adaptive systems: Method of averaging and persistency of excitation," *in IEEE Transactions on Automatic Control*, January 1987, vol. 32, no. 1, pp. 26–34.

[72] M. Yayla and A. T. Kutay, "Guaranteed exponential convergence without persistent excitation in adaptive control," *in 2016 UKACC 11th International Conference on Control*, 2016, pp. 1–6.

[73] M. M. Kogan and J. I. Neimark, "Locally optimal adaptive control without persistent excitation," Automatica, October 1996, vol. 32, no. 10, pp. 1463–1467. [74] T. Soderstrom and P. Stoica, "System Identification" Prentice Hall International, 1989.

[75] C. Ciochina, J. Paleologu, A. Benesty and A. Enescu, "Influence of the forgetting factor of the RLS adaptive filter in system identification," *in Proceedings of the International Symposiumon Signals, Circuits and Systems*, 2009.

[76] D. Rismayasari, E. Joelianta and D. Chaerani, "The implementation of robust - optimization based model predictive control to waste heat boiler," *International Conference on Instrumentation, Control and Automation*, Bandung, Indonesia, 2009, pp 184 - 189.

[77] I.R.F Guiamba, "Adaptive dynamic matrix control for a multivariable training plant." MSc Thesis, 2001.

[78] J. A. Rossiter, "Model-Based Predictive Control: A Practical Approach," CRC Press, 2003.

[79] R. Rouhani and R. K. Mehra, "Model Algorithmic Control; basic theoretical properties," Automatica, vol. 18, no. 4, pp. 401-414, 1982.

[80] J. Richalet, "Practique de la commande" Hermes, 1992.

[81] D. W. Clarke and C. Mohtadi, "Properties of Generalised Predictive Control; basic theoretical properties," Automatica, vol. 25, no. 6, pp. 859-875, 1987.

[82] E. Camacho and C. Bordons, Model Predictive Control. 2007.

[83] J. B. Rawlings and D. Q. Mayne, Model predictive control: theory and design. Madison, Wis: Nob Hill Pub, 2009.

[84] N.S. Shah, "Simulation of Model Predictive Control using Dynamic matrix Control Algorithm," MSc Thesis, California State University, December 2015.

[85] J. Lopez-Guede, B. Fernandez-Gauna, M. Grana, and F. oterino, “Model Predictive Control, Dynamic Matrix Control, Prediction Horizon,” *International Journal of Control Science and Engineering*, Spain, 2013.

[86] H. Maurice, “MPC and constrained systems.” [Online]. Available: https://heemels.tue.nl/research/mpc-and-constrained-systems. [Accessed: 20-Apr-2017].

[87] H. Bolandi, M. Rezaei, R. Mohsenipour, H. Nemati and S.M. Smailzadeh, “Attitude Control of a Quadrotor with Optimized PID Controller,” *Intelligent Control and Automation*, 2013, volume 4, no. 3, pp. 335–342.

[88] M. P. V. Sahul, V. Chander and T. Kurian, “A novel method on Disturbance Rejection PID Controller for Quadcopter based on Optimization algorithm,” *3rd International Conference on Advances in Control and Optimization of Dynamical Systems*, 2014, volume 47, pp. 192–199.

[89] T. Klopot and P. Skupin “Adaptive dynamic matrix control with interpolated parameters,” *20th International Conference on Methods and Models in Automation and Robotics (MMAR)*, 2015, pp. 683–688.

[90] J. Posada and M. Sanjuan, “Intelligent adaptive dynamic matrix control,” *The International Federation of Automatic Control*, 2008.

[91] S.N. Maiti, D.N. Saraf and N. Kapoor, “Adaptive dynamic matrix control of a distillation column with closed-loop online identfication,” *Journal of Process Control*, vol. 5, pp. 315–327, 1995.

[92] K.Y. Lee and U.C. Moon, “A boiler-turbine system control using a fuzzy auto-regressive moving average (FARMA) model,” *in IEEE T. Energy Conver.*, 2003 vol. 18, pp. 142–148.

[93] P. Lundstrom, J. H. Lee, M. Morari, and S. Skogestad, “Limitations of dynamic matrix control,” *Computers and Chemical Engineering*, 1995, vol. 19, no. 4, pp. 409–421.

[94] X. Chen, S. Li, J. Zhai and Q. Li, “Expert System Based Adaptive Dynamic Matrix Control for Ball Mill Grinding Circuit,” *Expert Syst. Appl.*, 2009, vol. 36, 716–723.

[95] M. Demircan, M.C. Camurdan and B.E. Postlethwaite, “On-Line Learning Fuzzy Relational Model Based Dynamic Matrix Control of an Openloop Unstable Process,” *Chemical Engineering Research and Design*, 1999, vol. 36, no. 5, pp. 421–428.

[96] S. Ramesh, M. Vijayakarthick, E. Sivaraman and S. Sathishbabu, "Implementation of Dynamic Matrix Control for Non-Linear System," International Research Journal of Engineering and Technology (IRJET), Mar 2016, vol. 3, no. 3, pp. 1397 -1402.

[97] J. Balakrishnan, "Control Systems Design Using Multiple Models, Switching, and Tuning," PhD thesis, Yale University, 1995.

[98] D. Borrelli, A. S. Morse, and E. Mosca, "Discrete-time supervisory control of families of two degrees-of-freedom linear set-point controllers," *IEEE Trans. on Automatic Control*, January 1998, vol. 44, pp. 178–181.

[99] A. S. Morse, "Supervisory control of families of linear set-point controllers — part 1: exact matching," *IEEE Trans. on Automatic Control*, October 1996, vol. 41, pp. 1413–1431.

[100] Young Sam Lee,Jinsuk Choi, Sugkil Seo and Yeong Sang Park, "Development of a Simulink Dynamic Matrix Control(DMC) Block for Use with an RCP System and Its Application to Motor Control," *In Proceedings of the 13th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2016)*, vol. 1, pp. 413-420.

[101] H. Ferreau, C. Kirches, A. Potschka, H. Bock and M. Diehl, "qpOASES: a parametric active-set algorithm for quadratic programming,". Mathematics Programming and Computer Science, vol. 6, pp. 327–363.

[102] Y. Lee, G. Gyeong, and J. Park, "QP Solution for the implementation of predictive control on microcontroller systems and its application method," *Journal of Institute of Control, Robotics and Systems*, 2014, vol. 20, no. 9, pp. 908–913.

[103] C. E. Garcia and M. Morari, "Internal model control, A unifying review and some new results," *Industrial and Engineering Chemistry Process Design and Development*, 1982, vol. 21, pp. 308-323.

[104] T. Sekara and M. Matausek, "Optimization of PID controller based on maximization of the proportional gain under constraints on robustness and sensitivity to measurement noise," *IEEE Transactions of Automatic Control*, 2009, vol. 54, no. 1.

[105] C. R. Cutler and B. L. Ramaker, "Dynamic Matrix Control - A Computer Control Algorithm," *American Institute of Chemical Engineers (AIChE) National Meeting*, Houston, TX, 1979.

[106] S. Kuznicki and D. Lee, "Pixhawk Pilot Support Package (PSP) user guide," MathWorks. Available: https://www.mathworks.com/ hardware-support/ pixhawk.html.

[107] S. Gomar, M. Mirhassani and M Ahmadi, "Precise digital implementations of hyperbolic tanh and sigmoid function," *50th Asilomar Conference on Signals, Systems and Computers*, 2016.

[108] W. Bylsma, "Approximating Smooth Step Functions Using Partial Fourier Series Sums," Army Tank Automotive Research Development and Engineering Center, Michigan, 2012.

Sample input-output flight data

This Appendix demonstrates a sample input-output flight data log. Altitude is in this case measured by a barometric sensor and is defined as height above sea level not ground.

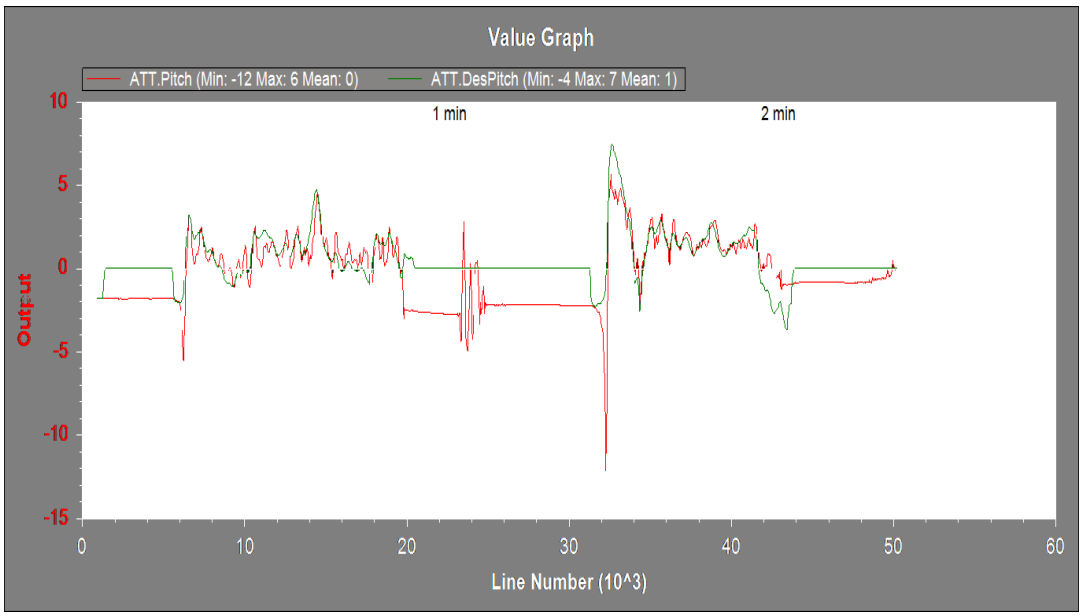

FIGURE A.1: Altitude experiment input-output data

Quadrotor model parameters

This Appendix demonstrates the simulation quadrotor model parameters.

$Ix = 7.5 * 10^{-3}$ Quadrotor moment of inertia around X axis

$Iy = 7.5 * 10^{-3}$ Quadrotor moment of inertia around Y axis

$Iz = 1.3 * 10^{-3}$ Quadrotor moment of inertia around Z axis

$Jr = 6.5 * 10^{-3}$ Total rotational moment of inertia around the propeller axis

$b = 3.13 * 10^{-3}$ Thrust factor

$d = 7.5 * 10^{-3}$ Drag factor

$l = 0.23$

$m = 0.55$

$g = 9.81$

Initial quadrotor step responses

This Appendix illustrates the initial quadrotor step responses before ADMC.

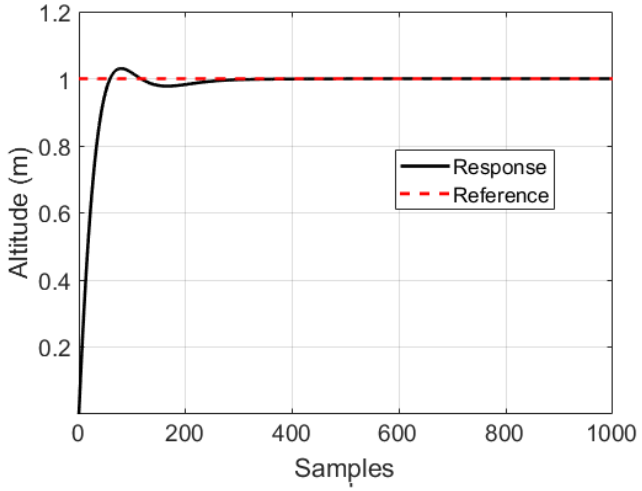

FIGURE C.1: Initial Z control step response

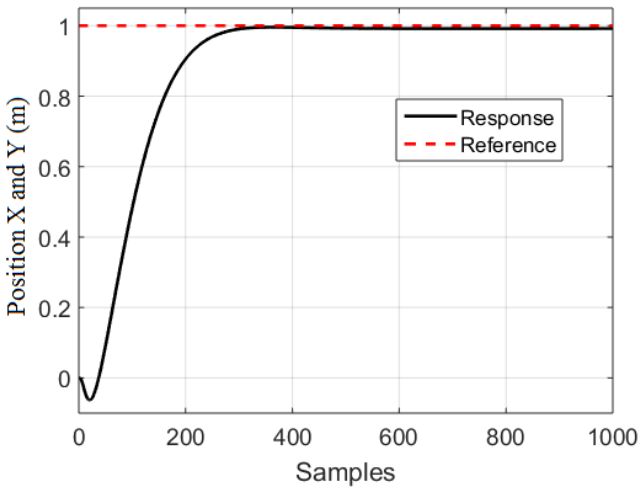

FIGURE C.2: Initial X and Y control step response

Pixhawk flight controller code generation

This Appendix details the code generation for the Pixhawk flight controller used in some parts of this research.

Signal limits

For the generated code to work correctly with the flight controller on the AXE quadrotor used for experiments, it is necessary to match the signal limits of the reference input and motor output. Motor output upper limits are given in the AXE manual, while reference input lower and upper limits are determined from RC calibration through the flight controller. It is established that the input limits for the AXE are as shown in Figure D.1 for the four channels of control. For code generation, it is necessary to configure Simulink to generate Pixhawk targeted code. This configuration is guided by instructions provide in [106].

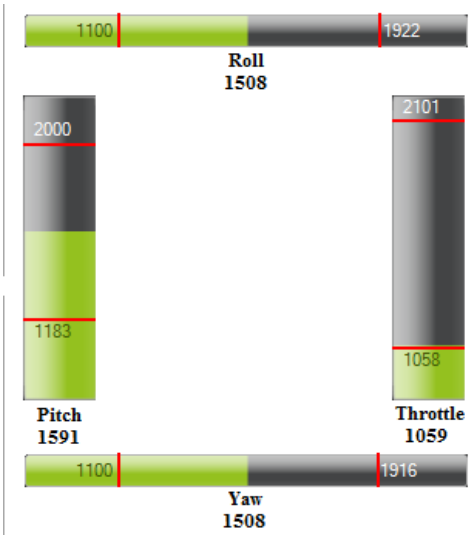

FIGURE D.1: Input signal limits

The relevance of this matching is important because in deploying the generated code onto Pixhawk, some auto calibration processes will be overwritten (as explained in the next Section).

Code generation

Using the tools provided by the Pixhawk Toolchain and the PSP, together with Simulink, the control code based on the desired controller can be generated and loaded onto Pixhawk. This code is in practice, loaded onto a micro-SD card which is then inserted into Pixhawk.

Will the generated code meet Pixhawk scheduling deadlines?

A serious problem that may arise in deploying the generated code is that if the algorithms are too heavy for Pixhawk, Pixhawk will shut down automatically. This matter is therefore considered seriously. To address this issue, the newer version of the support package must be used (2016). This version includes a feature to actually determine if the desired control algorithm will meet scheduling deadlines of Pixhawk. This is done by examining if the model generated code completes a task (call) on Pixhawk, in the specified sample time set by the Simulink model as shown in Figure D.2. If a task is not completed in the sample time, this is referred to as an over-run. In this way, there is guarantee that the flight controller will not shut down during flight.

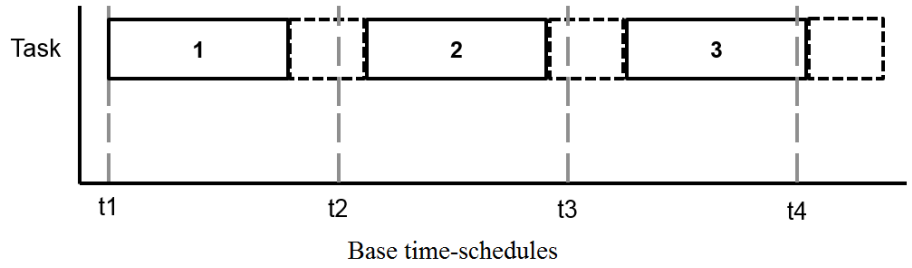

FIGURE D.2: Task over-run