Model Predictive Control of an Unmanned Quadrotor Helicopter: Theory and Flight Tests

Mahyar Abdolhosseini

A Thesis

in

The Department

of

Mechanical and Industrial Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science at

Concordia University

Montréal, Québec, Canada

September 2012

CONCORDIA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared,

By : **Mahyar Abdolhosseini**

Entitled : **Model Predictive Control of an Unmanned Quadrotor Helicopter: Theory and Flight Tests**

and submitted in partial fulfilment of the requirements for the degree of

Master of Applied Sciences (Mechanical Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. S. Narayanswamy

_____ Examiner
Dr. W-F. Xie

_____ Examiner
Dr. A.G. Aghdam External
Electrical and Computer Engineering

_____ Supervisor
Dr. Y. Zhang

_____ Co-Supervisor
Dr. C.A. Rabbath

Approved by:

_____
Dr. S. Narayanswamy, MASc Program Director
Department of Mechanical and Industrial Engineering

Date: _____

_____
Dean Robin Drew
Faculty of Engineering & Computer Science

ABSTRACT

Model Predictive Control (MPC) has been well established and widely used in the process control industry since years. However, due to dependability of its success on availability of high computational power to handle burden of online repetitive calculations, and existence of a precise mathematical model of the controlled plant, it has found less application in other areas of systems and control, specifically speaking when it comes to fast dynamics control systems featuring a highly elaborate plant.

Preceded by previous successful efforts made in the application of MPC to other areas of systems and control rather than process control, this thesis initiates employment of MPC in the unmanned aerial systems industry. To this end, the system of the quadrotor UAV testbed in the Networked Autonomous Vehicles Laboratory of Concordia University is chosen. A three dimensional autopilot control system within the framework of MPC is developed and tested through numerous flight experiments. The overall performance of the quadrotor helicopter is evaluated under autonomous fight for three flight scenarios of trajectory tracking, payload drop, robustness to voltage/current drop, and fault-tolerant control in the presence of faults induced by reduced actuator effectiveness. This has been achieved by the proper use of a model reduction technique as well as a fast optimization algorithm to address the issues with high computation, and incorporation of the integral action control in the MPC formulation to meet the offset-free tracking requirement. Both simulation and experimental results are presented to demonstrate success of the design.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisor Dr. Youmin Zhang. His guidance, encouragement, and patience made my thesis work a pleasant and rich experience. If I had never had access to the lab facilities he has been gathering gradually over years, I would never have made it through this project. He has always supported me, trough ups and downs and I really appreciate it.

Also, I would like to express my deepest gratitude to my co-supervisor Dr. Camille Alain Rabbath who supported me continuously for the whole period of my Master's studies. He has always been patient and let me take my time, whether I was getting good experimental results or not.

I would also like to thank two of my best friends, Dr. Abbas Chamseddine and Mr. Iman Sadeghzadeh. Abbas has always closely watched my work. He is deep in theory and teaches very well. Whether it was the design of an LQR controller or the real-time implementation of the designed autopilot control system, he was always next to me, ready to help. Iman is one of the rarest educated people I have ever met who has a feel for what he has been studying for years; he has absolutely practical ideas and this is highly appreciated, at any time anywhere. I would say, the success of this thesis would not have been possible without them.

I would like to thank Dr. Liuping Wang for his excellent book entitled *Model Predictive Control System Design and Implementation Using MATLAB*. Most of the materials in this thesis concerning model predictive control are derived from this book. In this thesis, almost half of the methodologies regarding discrete control design explained in the book are applied in practice without any major modification. Even though I was exposed to just half of this book, I would like to say that the ideas presented therein are completely practical

when it comes to practice and real time hardware implementation.

More than anyone else, I would like to thank my parents who brought me peace of mind, in any ways. This means all.

This work is partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) through a Strategic Project Grant and a Discovery Project Grant.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

1.1 Unmanned Aerial Vehicles (UAVs)

Unmanned quadrotor helicopters have become increasingly popular platforms for the study of Unmanned Aerial Vehicles (UAVs) from the control viewpoints. With the abilities such as hovering or vertical take-off and landing, quadrotor helicopters substantially extend the scope of potential civilian as well as military applications such as aerial reconnaissance, border patrol, life saving, and forest surveillance or fire fighting where it is highly risky for human pilots to intervene. Successful fulfilment of such missions is closely tied with existence of autopilot control systems. For the control of a quadrotor helicopter, various control techniques have been proposed. Initially starting with linear control algorithms such as LQR control [1] or PID control [2], linear methods are proved not to have a good performance for the nonlinear quadrotor system. The problem of nonlinear control design has been addressed using several methods such as feedback linearisation [3], sliding mode control [4] and back-stepping control [5]; nevertheless, among those nonlinear control methods, capability of explicitly dealing with operational constraints prevalent in a control system is yet hardly achievable. Fig. 1.1 depicts one of the quadrotor helicopters available at the Networked Autonomous Vehicles Laboratory of Concordia University. This

quadrotor helicopter is known as *Draganflyer*.

Figure 1.1: Draganflyer - Networked Autonomous Vehicles Laboratory of Concordia University

1.2 Model Predictive Control (MPC)

"Model Predictive Control, or Model-Based Predictive Control (MPC or MBPC as it is sometimes known), is the only advanced control technique–that is, more advanced than standard PID control–to have had a significant widespread impact on industrial process control" [6]. The capability of routinely dealing with equipment, performance and safety constraints allows for closer operation to a control system's limits, thus achieving the most profitable operation. In addition, expandability of the basic formulation to multi-variable plants without any major modification, simplicity of tuning, and the straightforwardness of its underlying idea, are certainly some of the main reasons that render this controller advanced.

Model predictive control was developed and used in the industry for nearly 20 years

before attracting much serious attention from the academic control community. An extensive study of the literature reveals that the era of model predictive control can be broken down into three decades of developments and achievements. The first decade is characterized by the fast-growing industrial adoption of the technology, primarily in the refining and petrochemical sectors. The second decade saw a number of significant advances in understanding the MPC from a control theoretician's viewpoint, while the third decade's main focus has been on the development of "fast MPC algorithms" [7].

Due to the specific structure of MPC which will be explained in the following sections, its successful implementation is highly dependent on availability of sufficient computational power. However, the constant increase in computational speed and power alongside the recent improvements in optimization algorithms which are the centrepiece of MPC, the use of this control technique is no longer bound to process control applications for which it was initially, almost exclusively envisioned [6]. In addition, recent advances in the MPC have led to its implementation onto faster dynamic systems and unstable plants, providing solutions to bring orders of magnitude improvement in the efficiency of the online computation so that the technology can be applied to systems and plants requiring very fast sampling rates, typical examples of which are frequently appeared in the field of aerospace design and innovation [8]. There has also been research into various model reduction techniques to minimize computational demands in order to render the MPC applicable to lightweight airborne platforms [9]. Furthermore, MPC strongly relies on a precise internal mathematical model of the plant under control. Since the real plant is invariably nonlinear, there exists always some degree of discrepancy between the mathematical model and the plant itself; therefore implementation of offset-free tracking control system with the MPC is hardly attainable unless measures are taken to address the issue of discrepancy.

By introduction of a new and sound model predictive control design framework, this study aims to partially address two main drawbacks of the MPC design, namely reliance

on:

- Availability of high computational power to handle burden of online repetitive calculations, and
- Existence of a precise mathematical model of the plant under control,

such that during the autonomous flight, an unmanned quadrotor helicopter with its fast elaborate dynamics can benefit from the numerous advantages that come along the proper use of this control technique.

To this end, firstly, a closed-loop prediction scheme will be offered for calculation of the predicted output y_p. This scheme will essentially reduce the computational load due to its structure. A new model reduction technique will be adopted based on some simplifying assumptions so that this closed-loop linear prediction scheme can be made use of. Also, as suggested by [9], in order to further reduce computational complexity thus execution time, it will be benefited from reduced number of prediction points that are not evenly placed along the prediction horizon–as required by the standard MPC variants. Secondly, as practised in some literature, it will be tried to meet the requirement of offset-free tracking by incorporating an integral-action controller in the outermost control loop so as to compensate for model uncertainties. Basically this control structure is a decentralized design which simply adds control inputs from the MPC and the integral algorithms. Although the steady state error can be eliminated by the integral controller's gain tuning, this control structure is incapable of constraint handling since the integrator dynamics is not included in the QP formulation [10]. Eventually, effort will be made to reformulate controller's structure to construct a centralized design. In contrast to the decentralized design, the new formulation does not simply add control inputs from the MPC and the integral algorithms but instead, the integral action is incorporated in the MPC formulation. This way, the steady state error is eliminated and the controller will be capable of constraint handling since dynamics of the integrator is included in the QP formulation.

The outline of the thesis is as follows. Chapter 1 presents an introduction to what an unmanned quadrotor helicopters is and how model predictive control can contribute to its applications. Chapter 2 deals with the idea of model predictive control and details two various formulations of the controller. Chapter 3 explains system software versus system hardware of Qball-X4, an unmanned quadrotor helicopter available at the Networked Autonomous Vehicles Laboratory of Concordia University. The overall performance of the quadrotor helicopter is evaluated under autonomous flight for three scenarios of trajectory tracking, payload drop, and robustness to voltage and current drop in Chapter 4. Finally, Chapter 5 draws the conclusion and outlines the future extensions of this study.

Chapter 2

Model Predictive Control

2.1 The Idea of "Predictive Control"

In what follows, the basic idea of model predictive control will be presented. For the sake of simplicity, discussion is confined to the control of a single-input single-output system. The idea and formulation set out herein will be applied to multi-input multi-output systems without loss of generality. Though a continuous version of this MPC design approach exists as well, a discrete-time setting will be discussed and applied.

As mentioned, a discrete-time setting is assumed, and the current time step is represented by k. A set-point trajectory which is the ideal or expected behavior of the control system is denoted by $s(t)$. Distinct from the set-point trajectory is the reference trajectory $r(t)$ that starts at the current output $y(k)$, and defines a second trajectory along which the plant should return to the set-point trajectory. Therefore, the reference trajectory determines an important behavioral aspect of the closed-loop control system. Although alternative definitions of the reference trajectory are possible, here an exponential reference trajectory is assumed with a time constant denoted by T_{ref} specifying the speed of the two trajectories' convergence or error reduction as in:

$$\varepsilon(k+i) = e^{-iT_s/T_{ref}}\varepsilon(k) \tag{2.1}$$

where

$$\varepsilon(k) = s(k) - y(k) \tag{2.2}$$

and T_s is the update rate of prediction. That is, the reference trajectory is defined to be:

$$r(k+i|k) = s(k+i) - \varepsilon(k+i) \tag{2.3}$$

$$= s(k+i) - e^{-iT_s/T_{ref}}\varepsilon(k) \tag{2.4}$$

There also exists an internal model which is employed to predict the behavior of the plant ahead of time over a prediction horizon starting at the current time. This predicted behavior is based on the assumed input trajectory $\hat{u}(k+i|k)$, $i = 0, 1, \ldots, H_p - 1$, that is to be applied over the prediction horizon, and the concept behind is to chose an input trajectory that results in the best predicted performance. It is assumed that the internal model is linear. In order to calculate the input trajectory, current output measurement $y(k)$ is required.

The elements of the input trajectory are selected in a way to bring the plant output $\hat{y}(k+i)$ to the corresponding value of the reference trajectory $r(k+i)$ at specific time intervals which may or may not be *evenly distributed*. In its simplest form, the input trajectory is chosen so that the plant output coincides with the reference trajectory at the end of the prediction horizon, namely $(k+H_p)$. In its most complex form, the input trajectory may be determined such that the plant output comes to the required reference trajectory at all sampling intervals $k+1, k+2, \ldots, k+H_p$ along the prediction horizon, introducing H_p coincidence points. For the case of a single coincidence point there are several input trajectories which achieve this. However, based on the criteria at hand one is chosen; for instance the input trajectory that minimizes the control effort may be preferred. In addition, with this wide possible range of selections, it is in fact recommended to impose some simple structure on the input trajectory. For example, the elements of the input trajectory may be allowed to vary over the first five steps of the prediction horizon, but to remain constant thereafter: $\hat{u}(k+4|k) = \hat{u}(k+5|k) = \cdots = \hat{u}(k+H_p-1|k)$. In this case there exist

five parameters to choose, namely $\hat{u}(k|k), \hat{u}(k+1|k), \hat{u}(k+2|k), \hat{u}(k+3|k), and \hat{u}(k+4|k)$. The idea of predictive control is illustrated in Fig. 2.1, schematically.

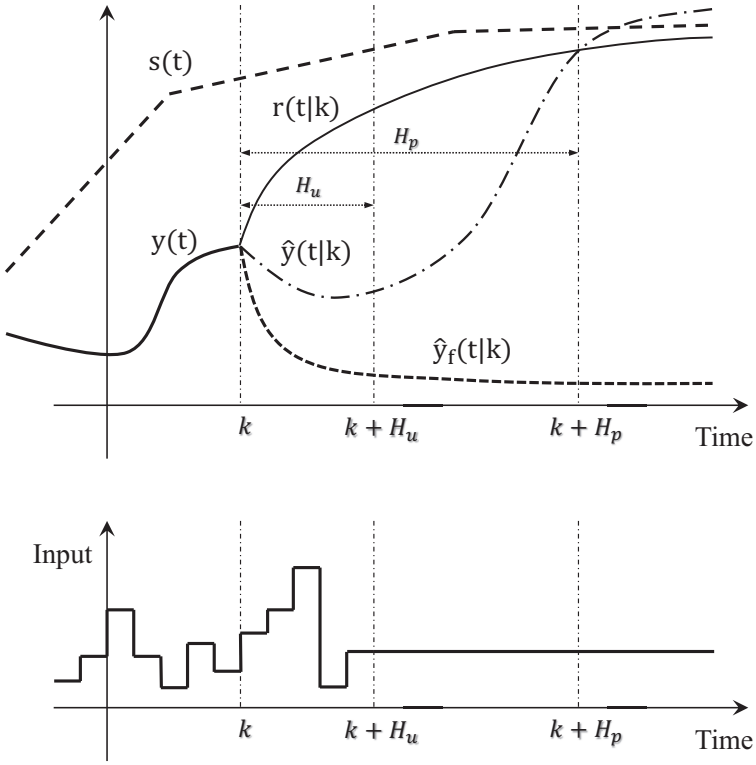

Figure 2.1: the Idea of Model Predictive Control

In practice, however, it is quite commonplace that there are more coincidence points than parameters to choose; that is to say, more equations to be satisfied than the number of available variables, and consequently impossible to find an exact solution. This implies lack of an exact future input trajectory capable of bringing the plant output to the reference trajectory at all coincidence points. That is the reason why some sort of approximate solution is sought, looking into a specific cost function. This can be a least-squared optimization problem, namely one that minimizes the sum of the squares of the error $\sum_i [r(k+i|k) - \hat{y}(k+i|k)]^2$, where i corresponds to the set of coincidence points [6, 11].

2.2 An Efficient Model Predictive Control Formulation

2.2.1 State Space Model Formulation

As the name implies, the centerpiece of a model predictive controller is a mathematical model of the real plant. This model should well represent behavioral characteristics of the control system under study, and is used to predict the free response of the plant; that is the response that would be obtained at the i^{th} coincidence point if the future input trajectory stays at the latest value having already been applied to the plant $u(k-1)$. To this end, for a state-space representation of the internal model, the current values of states or their estimations are needed. Assuming $S(i)$ to be the response of the internal model at some i^{th} coincidence point to a unit step function, as long as a linear time-invariant system is considered, the predicted output at the i^{th} coincidence point is:

$$\hat{y}(k+i|k) = \hat{y}_f(k+i|k) + S(i)\Delta\hat{u}(k|k) \tag{2.5}$$

where

$$\Delta\hat{u}(k|k) = \hat{u}(k|k) - u(k-1) \tag{2.6}$$

It is intended to achieve:

$$\hat{y}(k+i|k) = r(k+i|k) \tag{2.7}$$

Therefore, the optimal change of input is given by:

$$\Delta\hat{u}(k|k) = \frac{r(k+i|k) - \hat{y}_f(k+i|k)}{S(i)} \tag{2.8}$$

In a slightly complicated pattern for the input trajectory, the input is allowed to change over the first H_u steps of the prediction horizon, $\hat{u}(k|k), \hat{u}(k+1|k), \ldots, \hat{u}(k+H_u-$

$1|k)$; and remains constant thereafter, $\hat{u}(k+H_u-1|k) = \hat{u}(k+H_u|k) = \hat{u}(k+H_u+1|k) = \cdots = \hat{u}(k+H_p-1|k)$. This yields analogues results as obtained for the previous simpler input trajectory structure at the time step $k+P_i$ over the prediction horizon:

$$\begin{aligned}\hat{y}(k+P_i|k) = \hat{y}_f(k+P_i|k) + H(P_i)\hat{u}(k|k) + H(P_i-1)\hat{u}(k+1|k) + \ldots \\ + H(P_i-H_u+2)\hat{u}(k+H_u-2|k) \\ + S(P_i-H_u+1)\hat{u}(k+H_u-1|k)\end{aligned} \tag{2.9}$$

where $H(j) = S(j) - S(j-1)$ is the unit pulse response coefficient of the system after j time steps. The reason why pulse response coefficients appear in this expression rather than step response coefficients is that each of the input values $\hat{u}(k|k), \hat{u}(k+1|k), \ldots, \hat{u}(k+H_u-2|k)$ is to be applied for only one sampling interval. Only the last one, $\hat{u}(k+H_u-1|k)$, remains unchanged until step P_i, and its effect is therefore obtained by multiplying it by the step response coefficient $S(P_i-H_u+1)$. Since $H(j) = S(j) - S(j-1)$, Eq. (2.9) can be rewritten as:

$$\begin{aligned}\hat{y}(k+P_i|k) = \hat{y}_f(k+P_i|k) + S(P_i)\Delta\hat{u}(k|k) + S(P_i-1)\Delta\hat{u}(k+1|k) \\ + \cdots + S(P_i-H_u+1)\Delta\hat{u}(k+H_u-1|k)\end{aligned} \tag{2.10}$$

replacing equation (2.5).

Taking one step further by increasing the number of coincidence point, writing the same relation of a single coincidence point for each of the coincidence points and regrouping terms on both sides of the equation, the predicted output at the coincidence points in the matrix-vector form is:

$$\hat{Y} = \hat{Y}_f + \Theta\Delta\hat{U} \tag{2.11}$$

where

$$\hat{Y} = \begin{bmatrix} \hat{y}(k+P_1|k) \\ \hat{y}(k+P_2|k) \\ \vdots \\ \hat{y}(k+P_c|k) \end{bmatrix}; \quad \Delta\hat{U} = \begin{bmatrix} \Delta\hat{u}(k|k) \\ \Delta\hat{u}(k+1|k) \\ \vdots \\ \Delta\hat{u}(k+H_u-1|k) \end{bmatrix}$$

and

$$\Theta = \begin{bmatrix} S(P_1) & S(P_1-1) & \dots & S(1) & 0 & \dots & \dots & 0 \\ S(P_2) & S(P_2-1) & \dots & \dots & \dots & S(1) & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ S(P_c) & S(P_c-1) & \dots & \dots & \dots & \dots & \dots & S(P_c-H_u+1) \end{bmatrix}$$

replacing equation (2.5).

As stated earlier, it is intended to achieve (2.7). In the case of having more equations to be satisfied–corresponding to the number of coincidence points–than the number of available variables to be calculated, the solution of a least-squared optimization problem is sought to solve (2.8) for $\Delta\hat{U}$.

Having decided on the reference trajectory, a future input trajectory is easily calculated via (2.8). However, only the first element of that trajectory is to be applied as the input signal to the plant and the rest are neglected. Then the whole sequence of events being repeated one sampling interval later; that is, *output measurement*, *prediction*, and *input trajectory determination*. In the whole cycle of calculation, the prediction equations are used to determine the input trajectory, whereas output measurement is required to obtain the reference trajectory as well as the free response of the plant.

2.2.2 Realization of Constraints: Constrained Optimization

Predictive control is to be readily employed to respect constraints. Considering constraints on the inputs or outputs, the simple "linear least-squared" solution has to be replaced by a

"constrained least-squared" solution. Most formulations of predictive control assume linear inequality constraints; that is because even nonlinear constraints can be approximated by one or more linear constraints. For the case of constraints in the form of linear equalities, a quadratic programing problem evolves. This can be solved very reliably and relatively quickly by means of a number of efficient, computationally inexpensive optimization software available to date.

In practice, there are usually three types of constraints existent in a control system. Limitations that should be considered for actuator ranges available for the control effort, those of possible actuator slew rates, and constraints on the controlled variables. That is equivalent to:

$$a_1 < \Delta U(k) < a_2$$

$$b_1 < U(k) < b_2$$

$$c_1 < Y(k) < c_2$$

The following formulation represents the three types of constraints, respectively;

$$E \begin{bmatrix} \Delta U(k) \\ 1 \end{bmatrix} \leq 0; \quad F \begin{bmatrix} U(k) \\ 1 \end{bmatrix} \leq 0; \quad G \begin{bmatrix} Y(k) \\ 1 \end{bmatrix} \leq 0 \tag{2.12}$$

where

$$U(k) = \begin{bmatrix} \hat{u}(k|k)^T & \hat{u}(k+1|k)^T & \dots & \hat{u}(k+H_u-1|k)^T \end{bmatrix} \tag{2.13}$$

Assuming

$$G_c = \frac{1}{2}\Delta U^T \Phi \Delta U + \varphi^T \Delta U \tag{2.14}$$

as the cost function of a quadratic programing optimization problem with ΔU being its optimized solution, it is required to express all of the three types of constraints in terms of $\Delta U(k)$.

Suppose F has the form

$$F = \begin{bmatrix} F_1 & F_2 & \dots & F_{H_u} & f \end{bmatrix} \tag{2.15}$$

therefore, the second inequality of (2.12) can be written as:

$$\sum_{i=1}^{H_u} F_i \hat{u}(k+i-1|k) + f \leq 0 \tag{2.16}$$

since

$$\hat{u}(k+i-1|k) = u(k-1) + \sum_{j=0}^{i-1} \Delta\hat{u}(k+j|k) \tag{2.17}$$

the second inequality of (2.12) can be written as:

$$\begin{aligned} &\sum_{j=1}^{H_u} F_j \Delta\hat{u}(k|k) + \sum_{j=2}^{H_u} F_j \Delta\hat{u}(k+1|k) + \dots \\ &+ F_{H_u} \Delta\hat{u}(k+H_u-1|k) + \sum_{j=1}^{H_u} F_j u(k-1) + f \leq 0 \end{aligned} \tag{2.18}$$

By defining $\tilde{F}_i = \sum_{j=i}^{H_u} F_j$ and $\tilde{F} = [\tilde{F}_1, \tilde{F}_2, \dots, \tilde{F}_{H_u}]$, then the second inequality of (2.12) can be written as:

$$\tilde{F}\Delta U(k) \leq -\tilde{F}_1 u(k-1) - f \tag{2.19}$$

where the right-hand side of the inequality is a vector which is known at time k. The same methodology as discussed, can be applied to the third inequality of (2.12) so as to convert it into a linear inequality constraint on $\Delta U(k)$ [12].

Suppose G has the form

$$G = \begin{bmatrix} G_1 & G_2 & \dots & G_{P_c} & g \end{bmatrix} \tag{2.20}$$

therefore, the third inequality of (2.12) can be written as:

$$
\begin{aligned}
&G_1 Y_f(k+1|k) \sum_{i=0}^{H_u-1} S(P_1-i)\Delta\hat{u}(k+i|k) \\
&+ G_2 Y_f(k+2|k) \sum_{i=0}^{H_u-1} S(P_2-i)\Delta\hat{u}(k+i|k) + \dots \\
&+ G_{P_c} Y_f(k+P_c|k) \sum_{i=0}^{H_u-1} S(P_c-i)\Delta\hat{u}(k+i|k) + g \leq 0
\end{aligned}
$$

By defining $\tilde{G}_j = \sum_{i=0}^{H_u-1} S(P_j - i)$ and $\tilde{G} = [\tilde{G}_1, \tilde{G}_2, \dots, \tilde{G}_{P_c}]$, then the third inequality of (2.12) can be written as:

$$
\tilde{G}\Delta U(k) \leq -\sum_{j=1}^{P_c} G_j Y_f(k+j|k) - g \tag{2.21}
$$

where the right-hand side of the inequality is a vector which is known at time k. By transforming the first inequality of relations (2.12) into $W\Delta U(k) \leq w$ and assembling this with (2.19) and (2.21), the problem becomes that of

$$
min\ G_c = \frac{1}{2}\Delta U^T \Phi \Delta U + \varphi^T \Delta U \tag{2.22}
$$

subject to

$$
\begin{bmatrix} W \\ \tilde{F} \\ \tilde{G} \end{bmatrix} \Delta U \leq \begin{bmatrix} w \\ -\tilde{F}_1 u(k-1) - f \\ -\sum_{j=1}^{P_c} G_j Y_f(k+j|k) - g \end{bmatrix} \tag{2.23}
$$

2.3 An Integral-Action Model Predictive Control Formulation

There are three general approaches to predictive control design, each featuring a unique model structure. In the earlier formulation of model predictive control, finite impulse response and step response models received major attention. Soon after, they were found to be limited to stable plants and often required large model orders, typically ranging from 30 to 60 impulse response coefficients depending on the specific plant dynamics and choice of sampling Intervals. Transfer function models proved to offer a better representation of a plant comparably however, the transfer function model-based predictive control is often considered to be less effective in handling multi-variable plants. Recent years have seen the growing popularity of predictive control design using state-space methods, both in continuous time and discrete time. This is mainly due to simplicity of the design framework. In this section the structure of discrete-time Model Predictive Control with Integral action is discussed using the state-space formulation.

2.3.1 State Space Model Formulation with Embedded Integrators

As mentioned previously, model predictive control systems are designed based on a mathematical model of the plant. In this approach, the model to be used in the control system design is taken to be a state-space model. By using a state-space model, the current information required for predicting plant behaviour ahead of time is obtained through the state variable at the current time.

It is assumed that the underlying plant is described by:

$$x_m(k+1) = A_m x_m(k) + B_m u(k) \tag{2.24}$$

$$y(k) = C_m x_m(k) \tag{2.25}$$

where u is the control signal or input variable, y is the process output, and x_m is the state variable vector with assumed dimension $(n_1 \times 1)$.

To meet the offset-free tracking requirement, it is desired to slightly modify the model by embedding an Integral action in order to achieve the design purpose of *offset-free tracking*. In the general formulation of a state-space model there exists a direct term from the input signal $u(k)$ to the output $y(k)$ as in:

$$y(k) = C_m x_m(k) + D_m u(k)$$

However, due to the principle of receding horizon control, where a current information of the plant is required for prediction and control it has been implicitly assumed that the input $u(k)$ cannot affect the output $y(k)$ at the same time. Thus, $D_m = 0$ in the plant model. Taking a difference operation on both sides of (2.24) yields:

$$x_m(k+1) - x_m(k) = A_m(x_m(k) - x_m(k-1)) + B_m(u(k) - u(k-1))$$

Also by denoting the difference of the state and control variables by:

$$\Delta x_m(k) = x_m(k) - x_m(k-1)$$
$$\Delta u(k) = u(k) - u(k-1)$$

respectively, as the increments of the variables $x_m(k)$ and $u(k)$, the finite difference representation of the state-space equation is:

$$\Delta x_m(k+1) = A_m \Delta x_m(k) + B_m \Delta u(k) \tag{2.26}$$

where the input to the state-space model is $\Delta u(k)$. The next step is to connect $\Delta x_m(k)$ to the

output $y(k)$. To this end, a new state variable vector is chosen to be:

$$x(k) = [\Delta x_m(k)^T y(k)]^T$$

where superscript T indicates matrix transpose. Following the same procedure as before yields:

$$\begin{aligned} y(k+1) - y(k) &= C_m(x_m(k+1) - x_m(k)) \\ &= C_m \Delta x_m(k+1) \\ &= C_m A_m \Delta x_m(k) + C_m B_m \Delta u(k) \end{aligned} \tag{2.27}$$

Putting together (2.26) with (2.27) leads to the following state-space model:

$$\begin{bmatrix} \Delta x_m(k+1) \\ y(k+1) \end{bmatrix} = \begin{bmatrix} A_m & o_m^T \\ C_m A_m & 1 \end{bmatrix} \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix} + \begin{bmatrix} B_m \\ C_m B_m \end{bmatrix} \Delta u(k)$$

$$y(k) = \begin{bmatrix} o_m & 1 \end{bmatrix} \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix} \tag{2.28}$$

where $o_m = [0\,0\,\ldots\,0]$ contains n_1 zero entries. The triplet (A,B,C) is called the augmented model, which will be used in the design of predictive control.

Eigenvalues of the Augmented Model Considering a system of p inputs and q outputs, the characteristic polynomial equation of the augmented model is:

$$\rho(\lambda) = det \begin{bmatrix} \lambda I - A_m & {o_m}^T \\ -C_m A_m & (\lambda - 1) I_{q \times q} \end{bmatrix} = (\lambda - 1)^q det(\lambda I - A_m) = 0 \tag{2.29}$$

where the property that the determinant of a block lower triangular matrix equals the product of the determinants of the matrices on the diagonal has been used. Equation (2.29)

illustrates how the eigenvalues of the augmented model are the union of the eigenvalues of the plant model and the q eigenvalues, $\lambda = 1$. This means that there are q integrators embedded into the augmented design model. This is the means by which the integral action is incorporated into an MPC system.

Prediction of State and Output Variables Upon formulation of the mathematical model, the next step is to calculate the predicted plant output with the future control signal as the adjustable variables. Here, it is assumed that the current time is k_i and the length of the optimization window is N_p samples. It has been assumed that at the sampling instant k_i, the state variable vector $x(k_i)$ is available through measurement; this provides the current plant information. The future control trajectory is denoted by:

$$\Delta u(k_i),\ \Delta u(k_i+1),\ \Delta u(k_i+2),\ \ldots, \Delta u(k_i+N_c-1)$$

where N_c is called the control horizon dictating the number of parameters used to build the future control trajectory. With given information $x(k_i)$, the future state variables are predicted for N_p number of samples, where N_p is called the prediction horizon. The control horizon N_c is chosen to be less than (or equal to) the prediction horizon N_p.

Having denoted the future state variables by $x(k_i+m|k_i)$ as the predicted state variable at k_i+m with the given current plant information $x(k_i)$, based on the augmented state-space model (A,B,C), the future state variables are calculated sequentially using the set of

future control parameters as in:

$$\begin{aligned}
x(k_i+1|k_i) &= Ax(k_i)+B\Delta u(k_i)\\
x(k_i+2|k_i) &= Ax(k_i+1|k_i)+B\Delta u(k_i+1)\\
&= A^2x(k_i)+AB\Delta u(k_i)+B\Delta u(k_i+1)\\
&\vdots\\
x(k_i+N_p|k_i) &= A^{N_p}x(k_i)+A^{N_p-1}B\Delta u(k_i)+A^{N_p-2}B\Delta u(k_i+1)\\
&\quad +\cdots+A^{N_p-N_c}B\Delta u(k_i+N_c-1)
\end{aligned}$$

From the predicted state variables, the predicted output variables are, by substitution:

$$\begin{aligned}
y(k_i+1|k_i) &= CAx(k_i)+CB\Delta u(k_i)\\
y(k_i+2|k_i) &= CA^2x(k_i)+CAB\Delta u(k_i)+CB\Delta u(k_i+1)\\
y(k_i+3|k_i) &= CA^3x(k_i)+CA^2B\Delta u(k_i)+CAB\Delta u(k_i+1)+CB\Delta u(k_i+2)\\
&\vdots\\
y(k_i+N_p|k_i) &= CA^{N_p}x(k_i)+CA^{N_p-1}B\Delta u(k_i)+CA^{N_p-2}B\Delta u(k_i+1)\\
&\quad +\cdots+CA^{N_p-N_c}B\Delta u(k_i+N_c-1)
\end{aligned} \tag{2.30}$$

As it can be seen, all predicted variables are formulated in terms of current state variable information $x(k_i)$ and the future control movement $\Delta u(k_i+j)$, where $j=0,\ 1,\ \ldots,\ N_c-1$. Also by defining vectors Y and ΔU as:

$$\begin{aligned}
Y &= [y(k_i+1|k_i) \quad y(k_i+2|k_i) \quad y(k_i+3|k_i) \quad \ldots \quad y(k_i+N_p|k_i)]^T\\
\Delta U &= [\Delta u(k_i) \quad \Delta u(k_i+1) \quad \Delta u(k_i+2) \quad \ldots \quad \Delta u(k_i+N_c-1)]^T
\end{aligned}$$

equations (2.30) can be rewritten in a compact matrix form as:

$$Y = Fx(k_i) + \Phi\Delta U \tag{2.31}$$

where

$$F = \begin{bmatrix} CA \\ CA_2 \\ CA^3 \\ \vdots \\ CA^{N_p} \end{bmatrix}; \quad \Phi = \begin{bmatrix} CB & 0 & 0 & \dots & 0 \\ CAB & CB & 0 & \dots & 0 \\ CA^2B & CAB & CB & \dots & 0 \\ \vdots & & & & \\ CA^{N_p-1}B & CA^{N_p-2}B & CA^{N_p-3}B & \dots & CA^{N_p-N_c}B \end{bmatrix}$$

2.3.2 Seeking an Optimized Solution

Having defined a set-point signal $r(k_i)$ or a desired output, the objective of the model predictive controller at sample time k_i is to bring the predicted output as close as possible to the set-point signal, where it is assumed that the set-point signal remains constant over the prediction horizon, also referred to as the optimization window. This objective is then mathematically translated into finding a control signal vector ΔU such that a cost function containing an error function reflecting the discrepancy between the set-point signal and the predicted output is minimized. That is to say:

$$min\, J = (R_s - Y)^T (R_s - Y) + \Delta U^T \bar{R} \Delta U \tag{2.32}$$

where J denotes the cost function in which the first term is linked to the objective of minimizing the discrepancy just mentioned whereas, the second term refers to reducing the control effort while still achieving this objective, and:

$$R_s^T = [1\ 1\ \dots\ 1]_{1\times N_p} \cdot r(k_i)$$

is the data vector that contains information regarding the set-point signal. Also, in this expression $\bar{R}$ is a diagonal matrix in the form of $\bar{R} = r_w I_{N_c \times N_c} \quad (r_w \geq 0)$ where r_w acting on the control effort, is used as a tuning parameter for the desired closed-loop performance. For the cases of r_w being assigned relatively small values, the cost function (2.32) is interpreted as the situation where no matter how large the ΔU might be, the goal would be solely to make the error $(R_s - Y)^T (R_s - Y)$ as small as possible whereas, assignment of relatively large values is associated with situations where the controller would carefully consider how large the calculated control signal might be while cautiously reducing the error, keeping the control effort as low as possible and rendering the control system sluggish.

By substitution of the predicted output expressed by (2.31) the cost function J is expanded to:

$$J = (R_s - Fx(k_i))^T (R_s - Fx(k_i)) - 2\Delta U^T \Phi^T (R_s - Fx(k_i)) + \Delta U^T (\Phi^T \Phi + \bar{R}) \Delta U \quad (2.33)$$

The first term, though is a constant in the cost function, explains how the optimal solution of the control signal is tightly linked to the set-point signal $r(k_i)$ as well as the state variable $x(k_i)$ which is the most recent measurement taken and fed back, leading to a closed loop optimal control system.

2.3.3 Realization of Constraints: Constrained Optimization

Next is consideration of operational constraints that are frequently encountered in the design of control systems. This is where Model Predictive Control lends itself to; *the systematic handling of operational constraints*. Such constraints are usually presented as linear equalities and inequalities of the control and plant variables. In practice, there are three major types of constraints frequently encountered:

- Constraints on the Control Variable Incremental Variation
- Constraints on the Amplitude of the Control Variable

- Constraints on the Outputs or State Variables

From which the first two deal with the constraints imposed on the control variables $u(k)$, and the third deals with those on the outputs $y(k)$ or state variables $x(k)$. In the corresponding literature [12] it has been investigated how performance of a control system can deteriorate when a control signal produced by the controller reaches saturation limits of one or more of the actuators. On the other hand, with a small modification which accounts for incorporation of constraints, and acceptance of a small degree of performance degradation introduced to the control system, the previously talked about performance deterioration can be significantly eliminated. This is the motivation for consideration of constraints is a control system.

Having expressed operational constraints prevalent in a control system in terms of linear inequalities, it is required to relate them to the original Model Predictive Control problem. To this end, the set of equalities and inequalities reflecting constraints should be parameterized using the same parameter vector ΔU appeared within the cost function in the design of Model Predictive Control.

Constraints on the Rate of Change of a Control Signal There are constraints on the rate of change of the control variables $\Delta u(k)$, that is to say, on how big or small the control signal movements can be. A servomotor with the specification $0.1s/60^\circ$ provided by the manufacturer, will not travel 60° of its sweeping range in less than $0.1s$, whatever the the pulse width of the receiving PWM signal is. A control surface like an elevator in an airplane cannot sweep its whole deflecting range instantaneously; for instance, once receiving the command from an autopilot it takes some fraction of a second so that the surface returns to its neutral position from maximum upward deflection, then travels further aft to its maximum downward position, probably taking some other fraction of a second. Likewise, a telescopic hydraulic linear actuator cannot travel its effective range at an instant. No matter how fast or slow this actuator is, its time response is a certain value $t_s \quad (\kappa_{min} \leq t_s \leq \kappa_{max})$,

and may never extend beyond the specified range. This elapsed time is inevitable and should be considered and respected when designing a controller for a plant. For the case of constraints on the *Control Variable Incremental Variation*, this will be expressed by two inequalities:

$$-\Delta U \leq -\Delta U^{min}$$

$$\Delta U \leq \Delta U^{max}$$

In the matrix form, this becomes:

$$\begin{bmatrix} -I \\ I \end{bmatrix} \Delta U \leq \begin{bmatrix} -\Delta U^{min} \\ \Delta U^{max} \end{bmatrix} \tag{2.34}$$

where ΔU^{min} and ΔU^{max} are column vectors with N_c elements of Δu_{min} and Δu_{max}, respectively. This type of constraint can be equally used to implement one directional movement constraints on the control variable. As an example, if $u(k)$ can only increase and never decrease, the only way to impose this on a controller is by selecting $0 \leq \Delta u(k) \leq \Delta u^{max}$, provided that there exists a MPC based controller capable of dealing with constraints.

Constraints on the Amplitude of a Control Signal This is the most common type of constraint frequently faced with in practice. A valve cannot open more than 100% of its capacity; a control surface in an aircraft will not deflect more than a specific angle; a robotic arm does not reach a point out of its work space defined due to restrictions on its joints. These are some physical hard constraints and have to be respected. For the case of constraints on the *Amplitude of the Control Variable*, since:

$$\Delta u(k) = u(k) - u(k-1)$$

the control trajectory $u(k_i)$ $i = 1, 2, \ldots, N_c - 1$ can be expressed in terms of Δu as:

$$\begin{bmatrix} u(k_i) \\ u(k_i+1) \\ u(k_i+2) \\ \vdots \\ u(k_i+N_c-1) \end{bmatrix} = \begin{bmatrix} I \\ I \\ I \\ \vdots \\ I \end{bmatrix} u(k_i-1) + \begin{bmatrix} I & 0 & 0 & \ldots & 0 \\ I & I & 0 & \ldots & 0 \\ I & I & I & \ldots & 0 \\ \vdots & & & & \\ I & I & \ldots & I & I \end{bmatrix} \begin{bmatrix} \Delta u(k_i) \\ \Delta u(k_i+1) \\ \Delta u(k_i+2) \\ \vdots \\ \Delta u(k_i+N_c-1) \end{bmatrix}$$

or in a compact matrix form, with C_1 and C_2 corresponding to the appropriate matrices, then constraints on the *Amplitude of the Control Variable* are imposed as

$$-(C_1 u(k_i-1) + C_2 \Delta U) \leq -U^{min}$$
$$(C_1 u(k_i-1) + C_2 \Delta U) \leq U^{max}$$

where U^{min} and U^{max} are column vectors with N_c elements of u^{min} and u^{max}, respectively.

Constraints on an Output or a State Variable There exists also an operating range for a pant output. The temperature of a combustion chamber must not be less than a certain degrees Celsius if a combustion should happen, nor it should go beyond the melting point of materials used in its construction; the altitude of an airplane should be bound within a minimum and a maximum if the airplane intends to fly in a specified airway staying clear of other traffics flying around; the blood pressure of a human body should be maintained within a certain range if the blood is to circulate properly across the whole body. The same procedure applies in order to parameterrize the *Outputs* constraints using the same parameter vector ΔU as appeared within the cost function in the design of MPC, yielding:

$$-(Fx(k_i) + \Phi \Delta U \leq -Y^{min}$$
$$(Fx(k_i) + \Phi \Delta U) \leq Y^{max}$$

It is a common practice that output constraints be implemented as 'soft' constraints. This is achieved through introduction of a slack variable v_s into the upper and lower limits which define the operating range for the plant output. That is to say:

$$y^{min} - v_s \leq y(k) \leq y^{max} + v_s \tag{2.35}$$

There is am important reason behind why slack variables are introduced to the output constraints to render them soft constraints; output constrains, once being active, cause significant changes in both the control $u(k)$ and the incremental control $\Delta u(k)$ variables. This happens because the controller is trying to do its best not to violate output constraints. At the consequence of this the control and the incremental control variables violate their own constraints and this gives rise to a serious problem, saturations. In such circumstances, where the constraints on the control and the incremental control variables are more essential than that of the output, a big slack variable is introduced to the output constraints to avoid saturated actuators. As mentioned earlier, the third class of constraints which has been on plant outputs may equally be imposed on state variables–if they are measurable–or on the observer state variables. In this case, slack variables are employed in the same manner to transpose state variable constraints into soft constraints, hence preventing the same situation.

As the optimal solution will be obtained using a quadratic programming procedure, the constraints needed to be decomposed into two parts to reflect the lower limits, and the upper limits with opposite signs.Finally, the Model Predictive Control in the presence of hard constraints is proposed as finding the parameter vector ΔU that minimizes:

$$J = (R_s - Fx(k_i))^T (R_s - Fx(k_i)) - 2\Delta U^T \Phi^T (R_s - Fx(k_i)) + \Delta U^T (\Phi^T \Phi + \bar{R}) \Delta U \tag{2.36}$$

subject to the inequality constraints:

$$\begin{bmatrix} M_1 \\ M_2 \\ M_3 \end{bmatrix} \Delta U \leq \begin{bmatrix} N_1 \\ N_2 \\ N_3 \end{bmatrix} \tag{2.37}$$

where the data matrices are:

$$M_1 = \begin{bmatrix} -C_2 \\ C_2 \end{bmatrix}; \qquad N_1 = \begin{bmatrix} -U^{min} + C_1 u(k_i - 1) \\ U^{max} - C_1 u(k_i - 1) \end{bmatrix}$$

$$M_2 = \begin{bmatrix} -I \\ I \end{bmatrix}; \qquad N_2 = \begin{bmatrix} -\Delta U^{min} \\ \Delta U^{max} \end{bmatrix}$$

$$M_3 = \begin{bmatrix} -\Phi \\ \Phi \end{bmatrix}; \qquad N_3 = \begin{bmatrix} -Y^{min} + Fx(k_i) \\ Y^{min} - Fx(k_i) \end{bmatrix}$$

In principle, all the constraints are defined within the prediction horizon. This allows for their modification at the beginning of each optimization window. However, in order to reduce the computational load it is sometimes preferred to keep the constraints invariant with time and chose a smaller set of sampling instants–instead of all the future samples–at which the constraints are to be imposed [12].

The standard quadratic programming problem has been extensively studied in the literature [17], and this is a field of extensive investigation in its own right. The required numerical optimization solution for the Model Predictive Control is often regarded as an obstacle in the application of MPC due to limited computational power available, Nevertheless, Hildreth's Quadratic Programming Procedure proves to be computationally effective. This procedure was proposed for solving a group of problems collectively referred to as *Primal-Dual* to which the family of active set methods belongs. The idea of active set method is to define at each step of the algorithm a set of constraints, termed the working

set, which is to be treated as the active set. The working set is chosen to be a subset of the constraints that are actually active at the current point. The algorithm then proceeds to move on the surface defined by the working set of constraints to an improved point. In the active set methods, the active constraints need to be identified along with the optimization variable; therefore, an iterative procedure is required to solve the optimization problem with inequality constraints [18]. If the active set could be identified in advance, then the iterative procedure would be shortened; hence in the specific structure of the Hildreths QP procedure deployed in this work, it has been tries to address this pre-identification requirement. This will be further investigated in *Appendix B*.

2.3.4 State Estimation in Model Predictive Control

As mentioned earlier, in the design of model predictive control, it has been assumed that the information $x(k_i)$ is available at the time k_i. In other words, it is assumed that all the state variables are measurable. However, in reality and with most applications it happens quite often that not all state variables are measured or not all are available for measurement. One possible solution to address this problem is the use of a soft instrument to estimate the values of unknown state variables $x(k)$ based on the plant output measurement. This is commonly referred to as an observer. Observers are not necessarily to be employed for estimation of unknown state variables. In a noisy environment, a state observer can also act like a noise filter to reduce the effects of noise on the measurement of measurable state variables.

An observer is constructed based on the mathematical model of a plant. Here, construction of one will be detailed as explained in [12]. Assuming the plant model in the form of state space difference equations:

$$x_m(k+1) = A_m x_m(k) + B_m u(k) \tag{2.38}$$

This model can be used to calculate the state variable $\hat{x}_m(k)$, $k = 1,2,\ldots$ with an initial state condition $\hat{x}_m(0)$ and the input signal $u(k)$ as:

$$\hat{x}_m(k+1) = A_m\hat{x}_m(k) + B_m u(k) \tag{2.39}$$

Provided that the plant model is stable and the initial condition just substituted is nearly correct, this approach in fact would work after some transient time. However, this is an open-loop prediction and the prediction $\hat{x}_m(k)$ may not necessarily converge to $x_m(k)$. This will be further investigated. The error $\tilde{x}_m(k) = x_m(k) - \hat{x}_m(k)$ satisfies the difference equation:

$$\begin{aligned}\tilde{x}_m(k+1) &= A_m(x_m(k) - \hat{x}_m(k)) \\ &= A_m\tilde{x}_m(k)\end{aligned} \tag{2.40}$$

For a give initial error state $\tilde{x}_m(k) \neq 0$:

$$\tilde{x}_(k) = {A_m}^k\tilde{x}_m(0) \tag{2.41}$$

Therefore,

- If A_m has all the eigenvalues inside the unit circle, then the error system 2.41 is stable and $|\tilde{x}_m(k)| \to 0$ as $k \to \infty$, which means that the estimated state variable $\hat{x}_m(k)$ converges to $x_m(k)$. On the contrary, if A_m has one or more eigenvalues outside the unit circle, the error system 2.41 is unstable and $|\tilde{x}_m(k)| \to \infty$ as $k \to \infty$, which means that the prediction $\hat{x}_m(k)$ does not converges to $x_m(k)$. If A_m has one or more eigenvalues on the unit circle, the error state $|\tilde{x}_m(k)|$ will not converge to zero.

- In addition, for the case of a stable plant model A_m, there is no control on the convergence rate of the error $|\tilde{x}_m(k)| \to 0$, which is dependent on the location of the plant poles. If the plant poles are close to the origin of the complex plane, then the error

converges at a fast rate to zero; otherwise, the convergence rate could be slow.

To improve the estimation of $x_m(k)$, the use of a feedback principle where an error signal is deployed to improve the estimation is recommended, thus the observer is constructed using the equation:

$$\hat{x}_m(k+1) = A_m\hat{x}_m(k) + B_m u(k) + K_{ob}(y(k) - C_m\hat{x}_m(k)) \tag{2.42}$$

where K_{ob} is the observer gain matrix. In this formulation, the state variable estimate $\hat{x}_m(k+1)$ consists of two terms. The first term is the original model, and the second therm is the correction term based on the error between the measured output and the predicted output using the estimate $\hat{x}_m(k)$.

To choose the observer gain K_{ob}, the closed-loop error equation is examined. By substituting $y(k) = C_m x_m(k)$ into 2.42, with the definition of error state $\tilde{x}_m = x_m(k) - \hat{x}_m(k)$:

$$\begin{aligned}\tilde{x}_m(k+1) &= A_m\tilde{x}_m(k) - K_{ob}C_m\tilde{x}_m(k) \\ &= (A_m - K_{ob}C_m)\tilde{x}_m(k)\end{aligned} \tag{2.43}$$

This, with the given initial error $\tilde{x}_m(0)$ yields:

$$\tilde{x}_m(k) = (A_m - K_{ob}C_m)^k\tilde{x}_m(0) \tag{2.44}$$

Comparing the observer error response given by 2.44 with the open-loop prediction 2.41, it is apparent that the observer gain K_{ob} can be used to manipulate the convergence rate of the error. If there is only a single output, a commonly used approach is to place the closed-loop eigenvalues of the error system matrix $(A_m - K_{ob}C_m)$ at a desired location of the complex plane. This method is also referred to as *Pole Placement*.

Definition of Observability Assuming an unforced system described by:

$$\begin{aligned} \dot{x} &= A_{n\times n}x \\ y &= C_{m\times n}x \end{aligned} \tag{2.45}$$

the system is said to be completely observable if every state $x(t_0)$ can be determined from the observation of $y(t)$ over a finit time interval $t_0 \leq t \leq t_1$. The system is, therefore, completely observable if every transition of the state eventually affects every element of the output vector. In other words, if the system is completely observable, then given the output $y(t)$ over a time interval $0 \leq t \leq t_1$, $x(0)$ is uniquely determined. It has been shown that this requires the rank of the $nm \times n$ observability matrix to be n. The obserability matrix is:

$$\begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix} \tag{2.46}$$

The concept of observability is useful in solving the problem of reconstructing unmeasurable state variables from measurable variables in the minimum possible length of time [25].

Kalman Filter If the pair (A_m, C_m) is observable, then for the single-output system, as discussed, a pole assignment strategy can be used to determine K_{ob} such that the eigenvalues of the observer–i.e. that of the matrix $A_m - K_{ob}C_m$–are at the desired location. However, for a multi-output system, K_{ob} can be calculated recursively using a Kalman filter. To this end, it is assumed that:

$$\begin{aligned} x_m(k+1) &= A_m x_m(k) + B_m u(k) + d(k) \\ y(k) &= C_m x_m(k) + \xi(k) \end{aligned} \tag{2.47}$$

with the covariance matrices of d and ξ, respectively, defined by:

$$E\{d(k)d(\tau)^T\} = \Theta\delta(k-\tau)$$
$$E\{\xi(k)\xi(\tau)^T\} = \Gamma\delta(k-\tau)$$

where $\delta(k-\tau) = 1$ if $k = \tau$, and $\delta(k-\tau) = 0$ if $k \neq \tau$.

The optimal observer gain K_{ob} is solved recursively for $i = 0, 1, \ldots,$ using:

$$K_{ob}(i) = A_m P(i) C_m{}^T (\Gamma + C_m P(i) C_m{}^T)^{-1} \tag{2.48}$$

where

$$P(i+1) = A_m\{P(i) - P(i)C_m{}^T(\Gamma + C_m P(i) C_m{}^T)^{-1} C_m P(i)\} A_m{}^T + \Theta$$

and

$$P(0) = E\{[x(0) - \hat{x}(0)][x(0) - \hat{x}(0)]^T\}$$

Then, as $k \to \infty$, the steady-state solution of 2.48 guarantees that the eigenvalues of $A_m - K_{ob}(\infty)C_m$ are inside the unit circle, thus stable. It is emphasized that the iterative solution of 2.48 is not required in real time. The observer gain is calculated off-line for predictive control applications.

It is often the case that the covariance matrices Θ and Γ, corresponding to the characteristics of the disturbances, are unknown. Thus, in practice, Θ, Γ, and an initial $P(0)$ are chosen to calculate an observer gain K_{ob} by solving 2.48 iteratively until the solution converges to a constant matrix. Then, the closed-loop system obtained is analyzed with respect to the location of eigenvalues contained in $A_m - K_{ob}C_m$ i.e., the transient response of the observer, robustness, and effect of noise on the response are all investigated. Then, the elements of the covariance matrices are modified until a desired result is obtained. Such a trial-and-error procedure can be time consuming, and is one of the challenges faced with

when using Kalman-filter-based multivariable system design.

However, sometimes it is possible to specify a region in which the closed-loop observer error system poles should be and then enforce this in the solution. As proposed in [12] it is possible to design an observer whose closed-loop poles are bound to be inside a circle with a pre-specified radius α $(0 < \alpha < 1)$. Having defined $\tilde{x}(k) = x(k) - \hat{x}(k)$ as the error of the estimated state, then the observer error system is:

$$\tilde{x}(k+1) = (A_m - K_{ob}C_m)\tilde{x}(k) \tag{2.49}$$

Also, by performing the transformation $\hat{A}_m = \frac{A_m}{\alpha}$ and $\hat{C}_m = \frac{C_m}{\alpha}$ where $0 < \alpha < 1$, the transformed observer error system is:

$$\begin{aligned} \tilde{x}_t(k+1) &= (\hat{A}_m - \hat{K}_{ob}\hat{C}_m)\tilde{x}_t(k) \\ &= \frac{1}{\alpha}(A_m - \hat{K}_{ob}C_m)\tilde{x}_t(k) \end{aligned} \tag{2.50}$$

Solving the iterative equation 2.48 by using $\hat{A}_m$and $\hat{C}_m$ to replace A_m and C_m matrices, then the eigenvalues of $\hat{A}_m - \hat{K}_{ob}(\infty)\hat{C}_m$ are guaranteed to be inside the unit circle, yet stable. The calculated observer gain $\hat{K}_{ob}$ is then applied to the original observer system 2.49, leading to the closed-loop characteristic equation:

$$det(zI - (A_m - \hat{K}_{ob}C_m) = det(zI - (\hat{A}_m - \hat{K}_{ob}\hat{C}_m) \times \alpha) = 0 \tag{2.51}$$

Therefore, it concludes the study that the eigenvalues of $(A_m - \hat{K}_{ob}C_m)$ are the same as the eigenvalues of $(\hat{A}_m - \hat{K}_{ob}\hat{C}_m)$ multiplied by the factor α, which guarantees that the eigenvalues of the observer error system with $\hat{K}_{ob}$ be inside the circle of radius α. This procedure makes a direct connection to the dynamics of the observer through the choice of α; thus the trial-and-error procedure can be reduced to choose a suitable α along with Θ and Γ to achieve the desired closed-loop performance.

State Estimation and Model Predictive Control In the implementation of model predictive control state estimation is employed whenever one or more of the state variables $x(k_i)$ are not measured or available for measurement at time k_i. As mentioned in the previous section the state variable $x(k_i)$ is estimated via the observer structured as:

$$\hat{x}(k_i+1) = A\hat{x}(k_i) + B\Delta u(k_i) + K_{ob}(y(k_i) - C\hat{x}(k_i)) \tag{2.52}$$

It should be noted that in the structure of a model predictive controller, the control signal is Δu; that is the reason why in this formulation of an observer $u(k_i)$ is replaced with $\Delta u(k_i)$. Also, the matrices (A,B,C) are associated with the augmented model used in model predictive design. With the introduction of the estimated state(s) $\hat{x}(k_i)$ replacing $x(k_i)$, the predictive control law is slightly modified so that once again, it will be iterative calculation of ΔU while minimizing the cost function:

$$J = (R_s - F\hat{x}(k_i))^T(\bar{R}_s r(k_i) - F\hat{x}(k_i)) - 2\Delta U^T\Phi^T(R_s - F\hat{x}(k_i)) + \Delta U^T(\Phi^T\Phi + \bar{R})\Delta U \tag{2.53}$$

in which $\bar{R}_s$, F, Φ, $\bar{R}$, and ΔU are the same as before. Anonymously, the optimal solution is solved for as:

$$\Delta U = (\Phi^T\Phi + \bar{R})^{-1}\Phi^T(R_s - F\hat{x}(k_i)) \tag{2.54}$$

Summary In this section, the basic idea of MPC was discussed and two different formulations of the control technique were presented in a discrete-time setting, namely the efficient and the integral-action-incorporated formulations. The efficient formulation, executes essentially faster than the integral-action-incorporated type however, it is highly reliant on availability of a concise mathematical model of the plant. This issue has been addressed in the integral-action-incorporated formulation. Next section elaborates on the hardware and software structure of Qball-X4, an unmanned quadrotor helicopter on which flight experiments will be conducted throughout this research and development.

Chapter 3

Description of the Testbed

3.1 Dynamics of a Quadrotor Helicopter

A quadrotor helicopter consists of four rotors in a cross configuration. All the rotors axes of rotation are fixed and parallel and their propellers have fixed-pitch blades. These considerations imply that the structure is quite rigid and the only things that can vary are the propeller rotational speeds. The front and the rear propellers rotate clockwise, while the left and the right ones spin counter-clockwise. This configuration of pairs rotating in opposite directions eliminates the need for a tail rotor which is employed in the conventional helicopter configuration to counteract the reaction torque applied to the fuselage of a helicopter produced by the main rotor's rotation.

Even though a quadrotor has six degrees of freedom, there exists just four electrical motors to control motion of the vehicle, rendering the system under-actuated. Therefore, it is not possible to reach a desired setpoint for all the degrees of freedom, but a maximum of four. However, due to its structure, it is quite easy to chose the four best controllable variables and then decouple them to make its control easier. The four control variables are thus related to the four basic control movements which allow the helicopter reach a certain height and attitude.

In hovering condition, all the propellers have the same rotational speed to counterbalance the downward force due to gravity. Thus, the quadrotor performs stationary flight and no forces or torques push it away from its position.

Lift U_1 Lift is generated by increasing (or decreasing) all the propellers' rotational speed collectively by the same amount. This leads to a vertical force with respect to the body-fixed frame and raises or lowers the quadrotor. In this case, the speed of each propeller equals $\Omega_H + \Delta\omega_1$ with $\Delta\omega_1$ being a positive variable which represents an incremental lift to induce vertical motion. In what follows, by the proper use of circular arrows representing *direction* and *magnitude* of rotational speed of each propeller, it will be tried to illustrate three elements of the rotational motion of a quadrotor helicopter schematically, and explain how each of the three are induced.

Roll U_2 This command is provided by increasing (or decreasing) the left propeller's rotational speed and by decreasing (or increasing) that of the right one. It leads to a torque along the x_B axis which makes the quadrotor turn. The overall vertical thrust is the same as in hovering, hence this command leads only to a roll angle acceleration. Fig. 3.1 shows the roll command on a quadrotor sketch.

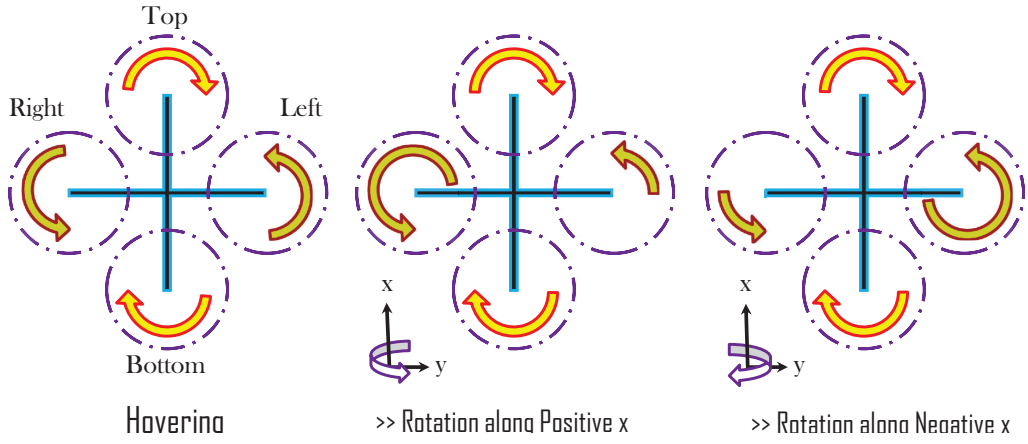

Figure 3.1: Transition from Hovering (on the Left) to Rolling Motion

Pitch U_3 This command is very similar to that of the roll and is provided by increasing (or decreasing) the rear propeller's rotational speed and by decreasing (or increasing) that of the front one. It leads to a torque along the y_B axis which makes the quadrotor turn. The overall vertical thrust is the same as in hovering, hence this command leads only to a pitch angle acceleration. Fig. 3.2 shows the pitch command on a quadrotor sketch.

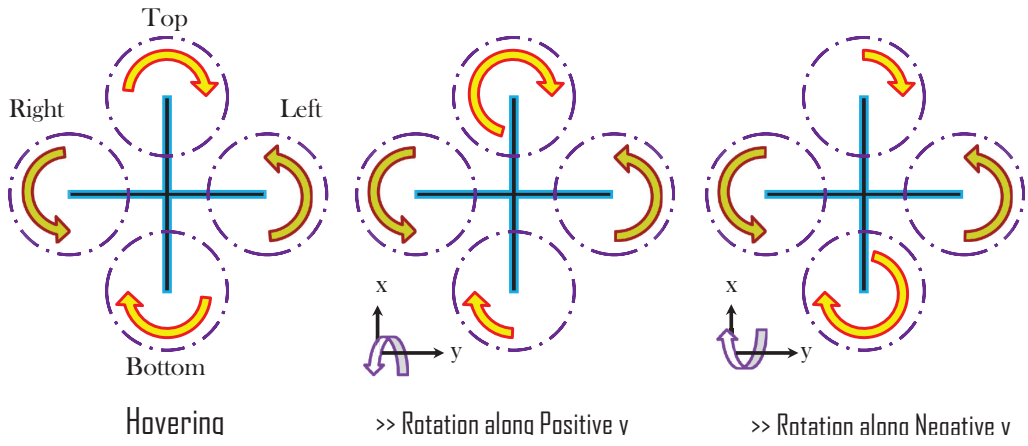

Figure 3.2: Transition from Hovering (on the Left) to Pitching Motion

Yaw U_4 This command is provided by increasing (or decreasing) the front and rear propellers' rotational speeds simultaneously and by decreasing (or increasing) that of the left and right propellers at the same time. It leads to a torque along the z_B axis which makes the quadrotor turn. The yaw movement is generated due to the fact that the left-right propellers rotate counter-clockwise while the front-rear ones rotate clockwise. Hence, when the overall torque is unbalanced, the helicopter spins around z_B in the opposite direction as that of the net torque induced by the unbalanced torques acting on the four rotors. The total vertical thrust is the same as in hovering, hence this command leads only to a yaw angle acceleration. Fig. 3.3 shows the yaw command on a quadrotor sketch.

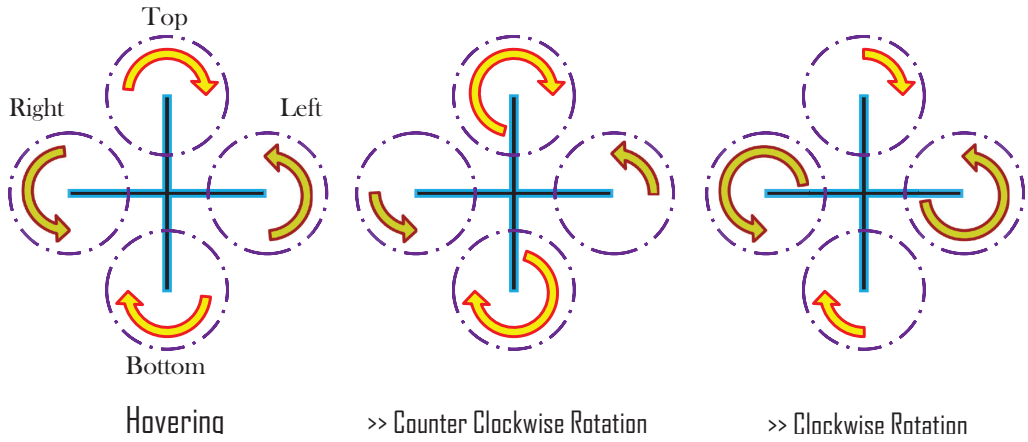

Figure 3.3: Transition from Hovering (on the Left) to Yawing Motion

3.1.1 Nonlinear Model of a Quadrotor Helicopter

Based on the balance of forces and moments as detailed in [14], equations of motion governing dynamics of a quadrotor helicopter with respect to an earth-fixed coordinate system are:

$$\ddot{x} = \frac{(\sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi)u_1 - K_1\dot{x}}{m} \tag{3.1}$$

$$\ddot{y} = \frac{(\sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi)u_1 - K_2\dot{y}}{m} \tag{3.2}$$

$$\ddot{z} = \frac{(\cos\phi\cos\theta)u_1 - K_3\dot{z}}{m} - g \tag{3.3}$$

$$\ddot{\phi} = \frac{u_3 l - K_4\dot{\phi}}{I_z} \tag{3.4}$$

$$\ddot{\theta} = \frac{u_2 l - K_5\dot{\theta}}{I_y} \tag{3.5}$$

$$\ddot{\psi} = \frac{u_4 c - K_6\dot{\psi}}{I_z} \tag{3.6}$$

where $K_i, \quad i = 1, 2, \ldots, 6$ are drag coefficients associated with the aerodynamic drag force, l is the distance between the center of gravity of the quadrotor and the center of each propeller, and c is the thrust-to-moment scaling factor. Note that the drag coefficients are negligible at low speeds. Also, I_x, I_y, and I_z represent the moments of inertia along

x, y, and z, respectively. The linear position $X_E = [x\ y\ z]^T$ of the body is determined by the coordinates of the vector connecting the origin of the earth fixed frame to that of the body fixed frame with respect to the earth fixed frame. The angular position (or attitude) $\Theta_E = [\phi\ \theta\ \psi]^T$ of the body is defined as the orientation of the body fixed frame with respect to the earth fixed frame. This is given by three consecutive rotations about the main axes to take the earth fixed frame into the body fixed frame. For this purpose, Euler angles are introduced: roll ϕ, pitch θ, and yaw ψ. As defined by convention:

- The earth fixed frame $(O_E\ x_E\ y_E\ z_E)$ is chosen as the inertial right-hand reference frame. x_E points toward the North, y_E points toward the West, z_E points upwards with respect to the earth and O_E is the center of this coordinate system. This frame is used to define the linear position X_E and the angular position Θ_E of the rigid body.

- The body fixed frame $(O_B\ x_B\ y_B\ z_B)$ is attached to the body being studied where O_B is generally chosen to coincide with its center of mass. This reference is right-hand too and it is used to define the linear velocity V_B, the angular velocity Ω_B, the forces F_B, and the torques τ_B.

The actuators of the quadrotor helicopter are brushless DC motors. The relation between the PWM input applied and the thrust produced by each is:

$$F_i = K_{motor}\frac{w_{motor}}{s + w_{motor}}u_{PWM} \tag{3.7}$$

where K_{motor} is a positive gain and w_{motor} represents the actuator bandwidth. For computational convenience the inputs to the system $u_i, \quad i = 1,\ 2,\ 3,\ 4$ are defined as:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -1 & 0 & 1 \\ -1 & 0 & 1 & 0 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} \tag{3.8}$$

Table 3.1 contains the nominal values of the quadrotor helicopter's system parameters.

Table 3.1: System Specifications

Parameter	m	I_x	I_y	I_z	l	c	K_{motor}	w_{motor}
Value	1.4	0.03	0.04	0.03	0.2	1	120	15
Unit	kg	$kg.m^2$	$kg.m^2$	$kg.m^2$	m	—	N	rad/s

3.1.2 Model Reduction to Minimize Computations

As stated earlier, due to the relatively high rate of update required for fast dynamic systems, success of predictive control in aerospace applications is highly dependent on the real-time computational power of the airborne computer. Since in almost all such applications the available onboard computational capacity is limited, partially due to weight considerations, any effort to reduce the burden of calculations is crucial to render application of the MPC to aerial systems–specifically unmanned vehicles–feasible.

To this end, it has been tried to decouple the six-degree-of-freedom equations of motion governing dynamics of the quadrotor so that the system is described by three plus one second-order differential equations, in which:

- The translational longitudinal displacement x is coupled with the rotational pitching motion θ,
- The translational lateral displacement y is coupled with the rotational rolling motion ϕ, and
- The translational vertical displacement along the normal axis z is treated separately and independently of the other two.

That is to say:

$$\ddot{x} = \frac{u_1 \sin\theta}{m}; \quad \ddot{\theta} = \frac{u_2 l}{I_y} \tag{3.9}$$

$$\ddot{y} = \frac{u_1 \sin\phi}{m}; \quad \ddot{\phi} = \frac{u_3 l}{I_x} \tag{3.10}$$

$$\ddot{z} = \frac{u_1}{m} - g; \quad \ddot{\psi} = \frac{u_4 c}{I_z} \tag{3.11}$$

This way, dimensions of the system matrices involved in the iterative calculations of predictive control including that of the optimization over a single time step–lasting for a fraction of a second–will be of the order of one-third or less; otherwise, direct consideration of a six-DOF motion corresponding to a quadrotor helicopter includes matrices of the order of fourteen (two corresponding to each degree of freedom plus those of DC motors). This individual treatment of the modes of motion greatly affects the execution time of onboard calculation. Also, regarding the yawing motion ψ, it has been assumed that a zero yaw angle is maintained at all times; this can be achieved by integration of a separate reaction-wheel mechanism–apart from the four DC motors–to take over control of the yawing motion.

With this new subset of equations, $\sin\theta$, $\sin\phi$, and $\frac{u_1}{m}g$ will be taken as manipulated variables or inputs of their corresponding equations (3.9–3.11). That is to say, u_1 is initially calculated by means of the third equation of (3.11) written for steady and level flight. Then this value is substituted in both the first equations of (3.9 and 3.10) as constant (over the prediction horizon), remaining $\sin\theta$ and $\sin\phi$ as the only manipulated variables. Next, the new versions of equations are discretized with a proper discretization time step, preserving dynamics of the quadrotor system. This rate can vary from one equation to the other depending on how agile the system acts along that axis.

3.1.3 Validation of the Simplified Decoupled Model vs. the Elaborate Coupled Model

In the previous section, based on some simplifying assumptions, a model reduction technique was used. However, the obtained decoupled model holds as long as those underlying simplifying assumptions are met; that is to say, the pitch angle as well as the roll angle are maintained within the vicinity of zero or thereabouts at all times. In other words, there is

a flight envelope inside which the quadrotor is bound to stay over the course of a flight, if the simplified decoupled model is to be used.

As stated, in order to arrive at the simplified decoupled equations of motion it is required to keep the rotational angles–roll, pitch, and yaw–as small as possible. But this is a qualitative image of the requirement. However, for the purpose of controller design this requirement should be precisely specified quantitatively as well. The controller that is designed based on the simplified model will not be functioning properly once the plant passes across or violates the boundaries of the pitch and roll angles determined to be respected for the validity of the employed model reduction technique. This is also referred to as flight envelope.

In this section, instead of conducting a set of simulations to determine the flight envelope, a single simulation is set up to reveal the validity range of the decoupled model throughout a flight. In this flight test the quadrotor is guided through a series of consecutive square trajectories of increasing sides. By increasing the sides of square trajectories, setpoint changes will be gradually increased as the dimensions of the square trajectories become bigger and bigger. In order to accommodate such abrupt changes of setpoint, the controllers output input signals of increasing amplitude as well. Since the controllers' outputs/the input signals to the plant are $\sin\theta$ and $\sin\phi$, soon their values will reach a point beyond which the decoupled model does not conform to the coupled full order model. This point should be marked as the bottom-line of design.

As suggested in Fig. 3.4, if the quadrotor receives a setpoint variation of 2 meters or more along the longitudinal axis, the longitudinal controller will output a pitch angle greater than 0.2618 rad (15 degrees) to accommodate such a setpoint change and makes the quadrotor tilt 15° either forwards or backwards, accordingly. This is the maximum acceptable change of pitch angle, if the decoupled simplified equations are to be used for the purpose of design. The lateral dynamics exhibits less sensitivity to variations in the roll angle. This is illustrated in Fig. 3.5. As the yaw angle does not contribute much to the cross

coupling of equations of motion, its variations will be neglected in the process of controller design. In addition, it has been assumed that a separate controller is employed to maintain a zero yaw angle essentially at all times; this is pretty manageable in practice.

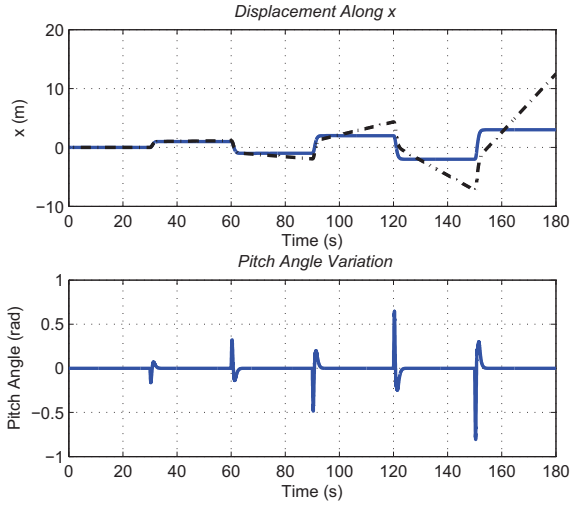

Figure 3.4: Validation of the Simplified Decoupled Model Along x

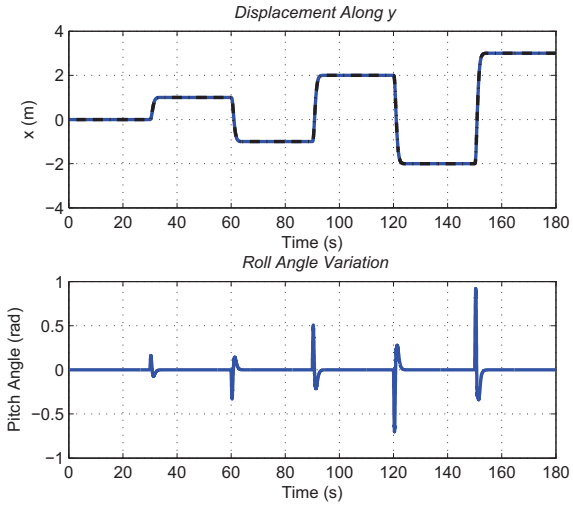

Figure 3.5: Validation of the Simplified Decoupled Model Along y

Therefore, a maximum of 2-meter setpoint change of translational longitudinal or lateral motion corresponding to the 0.2618 rad (15 degrees) change of either pitch or roll angle specifies boundaries of the pertinent flight envelope. However, this should not be

interpreted as an operational restriction for the developed MPC control system, yet a shortcoming of all other controllers, but not MPC. As mentioned previously, MPC is one of the rarest control techniques that can explicitly deal with operational constraints. Therefore, once the boundaries of $\sin\theta$ and $\sin\phi$ (yet θ and ϕ) are given to the controller as constraints on the manipulated variable or U, setpoint variations of whatever magnitude may be applied to the quadrotor helicopter. That is possible because the constrained MPC controller will never output a control signal less than -0.2618 rad (-15 dgrees) or greater than +0.2618 rad (+15 dgrees), maintaining $-15^\circ < \theta < +15^\circ$ and $-15^\circ < \phi < +15^\circ$ at all times. This is what distinguishes MPC from other controllers. That is illustrated in Fig. 3.6. In spite of abrupt setpoint variations of significant amplitude the developed controller keeps the quadrotor on the trajectory.

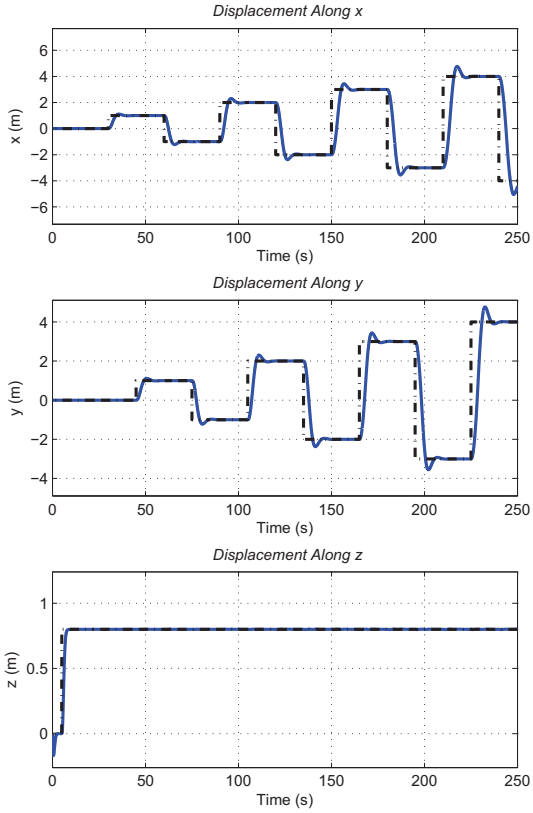

Figure 3.6: Abrupt Setpoint Variations of Great Amplitude - Constrained MPC

3.2 Qball-X4: Hardware vs. Software

The quadrotor UAV available at the Network Autonomous Vehicle (NAV) Lab in the Department of Mechanical and Industrial Engineering of Concordia University is the Quanser's Qball-X4 as shown in Fig. 3.7.

Figure 3.7: The Qball-X4 quadrotor UAV (Quanser, 2010)

The quadrotor UAV is enclosed within a ball-shaped protective carbon fiber cage to ensure safe operation. Generally speaking, this quadrotor helicopter platform is suitable for a wide variety of UAV research and development applications. This innovative rotary-wing vehicle is propelled by four DC motors fitted with 10 inch propellers. The entire system is enclosed within a spherical protective carbon fiber cage of 68 cm.

The Qball-X4's proprietary design ensures safe operation and opens the possibilities for a variety of novel applications. For instance, the protective cage is a crucial feature since this unmanned aerial vehicle is designed mainly for use in an indoor environment of a laboratory where there are typically many close-range hazards (including other vehicles) and personnel doing flight tests with the system. The cage gives the Qball-X4 a decisive advantage over other vehicles that would suffer significant damage if contact occurs between the vehicle and an obstacle. To have onboard sensor measurements and drive the motors, the Qball-X4 utilizes Quanser's onboard avionics data acquisition card

(DAQ), the HiQ, and the embedded single-board computer, Gumstix. The HiQ DAQ integrates a high-resolution Inertial Measurement Unit (IMU) and an avionics Input/Output (I/O) card designed to accommodate a wide variety of research applications. In addition, the onboard flight computer's open-architecture hardware and extensive Simulink blocksets provide users with powerful controls development tools.

QuaRC, Quanser's real-time control software, the interface to the Qball-X4 in MATLAB/Simulink environment, allows researchers and developers to rapidly develop and test controllers on actual hardware through the MATLAB/Simulink interface. QuaRC can target the Gumstix embedded computer automatically, generating the code and executing the designed controllers onboard the vehicle. In other words, the controllers are developed in Simulink with QuaRC on the host computer. Next, these models are coded, compiled into executable codes, and eventually uploaded on the target (Gumstix) seamlessly [15]. The open-architecture QuaRC and extensive Simulink blocksets provide users with powerful control development tools. The communication diagram as well as the Qball-X4 system configuration are shown in Fig. 3.8.

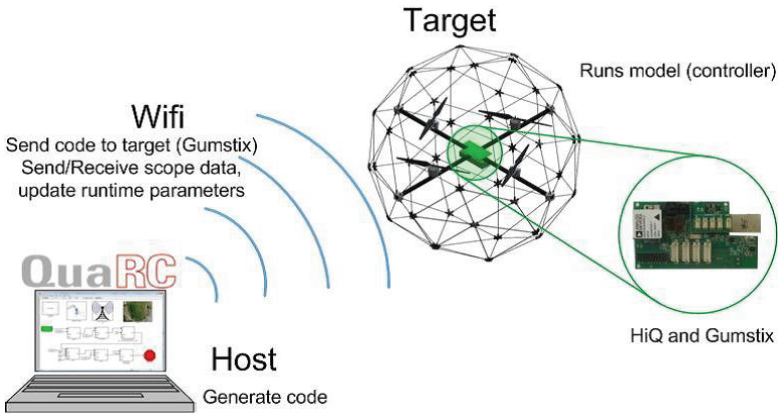

Figure 3.8: Qball-X4 communication hierarchy and communication diagram

During flights, while the controller is executing on the Gumstix, users can tune parameters in real time and observe sensor measurements from a host ground station computer (PC or laptop). System's main components include:

- Qball-X4: as explained previously;
- HiQ: QuaRC aerial vehicle data acquisition card (DAQ);
- Gumstix: The QuaRC target computer. An embedded, Linux-based system with the QuaRC runtime software installed;
- Batteries: Two 3-cell, 2500 mAh Lithium-Polymer batteries; and
- Real-Time Control Software: The QuaRC-Simulink control system development software.

OptiTrack Motion Tracking System for Localization As for any other moving robot demonstrating autonomous motion, decision making on where to go, in which direction to head, and with what speed to move is strongly dependent on the information regarding current position of the vehicle as well as its orientation.

Based on the environment in which a quadrotor helicopter is supposed to work, whether indoor or outdoor, the area to be covered during a flight, and the precision called for, different motion tracking systems can be appropriately competent to be used for navigation purposes. Even though such a quadrotor helicopter, once being developed, is supposed to fulfill an outdoor mission, for the time being, i.e. during the process of control algorithm development and performance evaluation, much of a testbed characteristics are required other than an industrial solution; in other words, flight in the environment of a laboratory for the purpose of research and development needs extensive accuracy rather than large coverage which is of concern for the finished unmanned quadrotor helicopter. That is the reason why criteria have been narrowed down to a system of indoor positioning cameras rather than other solutions like the Global Positioning System (GPS).

A set of three or more V100:R2 cameras which offers integrated image capture, processing, and motion tracking in a compact package constitute the OptiTrack's optical motion tracking system. The capability of customizing cameras with user-changeable M12

lenses, and OptiTrack's exclusive Filter Switcher technology has let V100 cameras deliver one of the world's premier optical tracking value propositions. Each V100:R2 camera is capable of capturing fast moving objects with its global shutter imager at 100 FPS capture speed. By maximizing its 640×480 VGA resolution through advanced image processing algorithms, the V100:R2 can also track markers down to sub-millimeter movements with maintainable accuracy [16].

A variety of V100:R2 settings are customized with any of OptiTrack's software applications such as the one employed in this study, i.e. *Tracking Tool*, for greater control over what cameras capture and what information they report to the personal computer set up as the ground station. Available settings include: image processing type, frame rate, exposure, threshold, illumination, filter switching, and status LED control. Some OptiTrack's software applications like the Tracking Tool interface with MATLAB, under manipulation of specific blocks inside MATLAB/Simulink within the library of QUARC. QUARC is a built-in blockset that integrates with Simulink, provided by the Mathworks Company. This has been explained in the previous section. Fig. 3.9 illustrates one of the six cameras employed constituting the system of OptiTrack.

Figure 3.9: The OptiTrack System – Camera #2

Summary In this section, equations of motion governing key dynamics of a quadrotor helicopter were presented. Based on the underlying assumption that MPC can keep both pitch and roll angles bounded by a tight limit maintaining smooth flight at all times, a model reduction technique was practiced. Validity analysis of the reduced model versus the real plant governed by the highly coupled original equations of motion were done to illustrate effectiveness of simplifications made. Eventually, the basic structure of the specific quadrotor helicopter under study, Qball-X4, was explained both in terms of hardware and software. In the following section, various stages of the autopilot control system design, simulation and flight test experiments, will be detailed.

Chapter 4

Development of the Autopilot

4.1 Phase 0: Efficient MPC Design

In this section, simulation and experimental results corresponding to the efficient MPC design introduced primarily due to its highly reduced computational demand are presented. In Chapter Two, Section Two, a fast prediction scheme was introduced within the framework of efficient model predictive control in order to present a fast MPC in which the burden of calculation has been highly reduced. However, this has been achieved at the expense of lost robustness against model uncertainties. Even small amounts of uncertainties added to system model, adversely effects offset-free tacking capability of the overall control system. This is illustrated in Fig. 4.1, schematically. Therefore, here it is intended to demonstrate the advantage as well as the disadvantage of the prediction scheme just proposed.

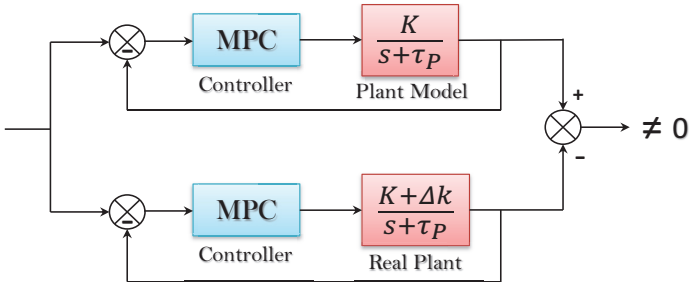

Figure 4.1: the Effect of Model Uncertainties on the Efficient Formulation

4.1.1 Simulation Results

As mentioned, this is a flaw in the prediction scheme set forward within the framework of efficient MPC formulated in Chapter Two, Section One. This deficiency should be fully recognised so that the boundaries of the efficient PMC design applications are set behind this deficiency that introduces some limitation to its use.

To this end, Qball-X4 simulation model has been given a rectangular trajectory to follow. A set of two experiments are set up. In one scenario, the designed controller is connected to a plant which is mathematically identical to the internal model of the MPC controller, whereas in the second one, the mass of the quadrotor helicopter is increased by 10%, such that some degree of discrepancy between the internal model and plant is added to the plant, rendering the plant different from the internal model. Fig. 4.2 refers to the first scenario in which the designed controller is connected to a plant identical to the internal model of the MPC controller, whereas Fig. 4.3 shows vulnerability of the design to a small 10% value of discrepancy introduced to the control system in terms of mass.

The time response of the efficient MPC algorithm is satisfactory, explained by time domain performance indices such as rise time, settling time, and overshoot. The run time of the non-real-time simulation running on a desktop computer featuring an Intel Core(TM) 2Duo CPU, 2.20GHz processor shows that it executes as fast as a PID controller does; and this is promising. However, its reliance on existence of a detailed precise mathematical model of the plant under control, restricts its use to control system applications for which such an elaborate model is derived, available for design. This does not account for the majority of applications. That is why a means is desperately sought to eliminate the need for an accurate model of the plant; that is incorporation of an integral action control into the design. Depending on how an integral action is incorporated into the structure of an MPC controller, two designs are imaginable, centralised design versus decentralized design. Fig. 4.4 compares the two structures, schematically.

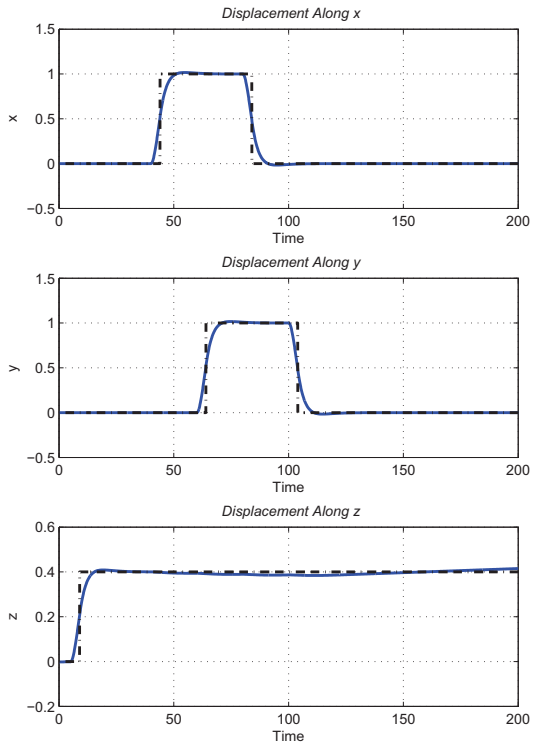

Figure 4.2: Performance of the Efficient MPC upon Existence of a Precise Mathematical Model

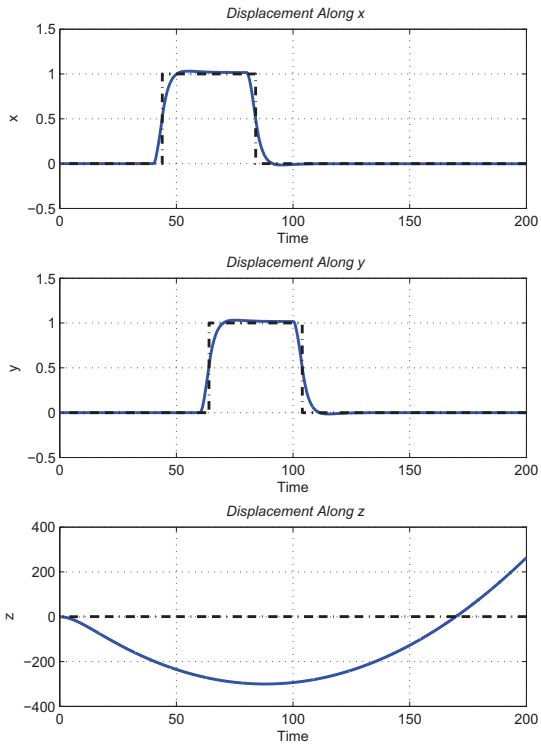

Figure 4.3: Performance of the Efficient MPC upon lack of Existence of a Precise Mathematical Model

4.1.2 Experimental Testing Results

Next is implementation of the decentralized controller on the Qball-X4 to assess performance of the approach. This is done by putting in parallel the developed controller with the baseline controller of the Qball-X4 which is a PID controller for height control. Having designed the control system in the environment of QuaRC for a single degree of freedom (height), the altitude-hold controller is built and uploaded to the onboard flight computer to take control of the vehicle. The proposed efficient MPC successfully controls the vehicle along a rectangular trajectory, as shown in Fig. 4.5. In this flight test, designed controller's parameters are as specified in Table 4.1.

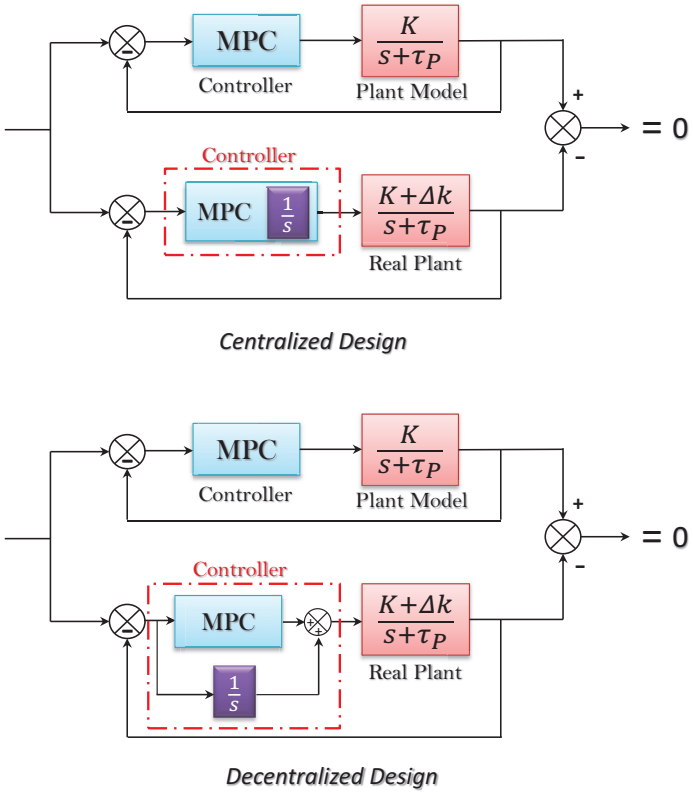

Figure 4.4: Centralized Design vs. Decentralized Design

Table 4.1: Controller's Design Parameters

Parameter	Value
T_{ref}	5
T_s	$0.01T_{ref}$
Prediction Horizon	80
Discretization Rate	0.05

Comparing with the simulation result presented in Fig. 4.2, there are some small differences due to the effects of measurement noises and disturbances added on the Qball-X4 during flights in the experimental testing environment. Basically this control structure is a decentralized design which simply adds control inputs from MPC and Integral control algorithms. Although the steady state error can be eliminated by Integral gain tuning, this control structure is incapable of constraint handling since the integrator dynamics is not included in the QP formulation [10]. As explained in Chapter Two, Section Three, effort has been made to reformulate controller's structure to construct a centralized design. In

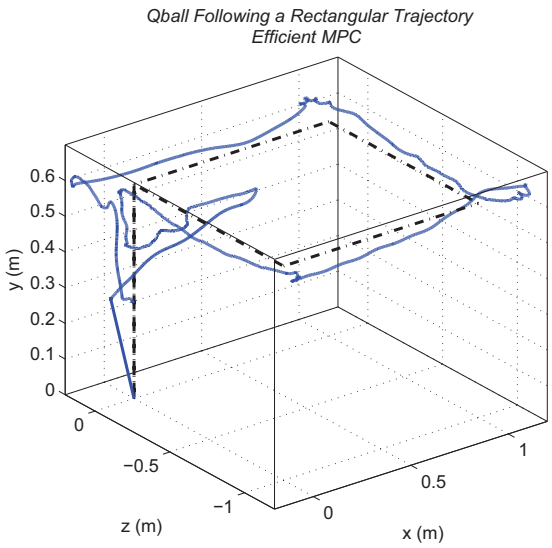

Figure 4.5: Qball-X4 Real-time Performance of Developed MPC

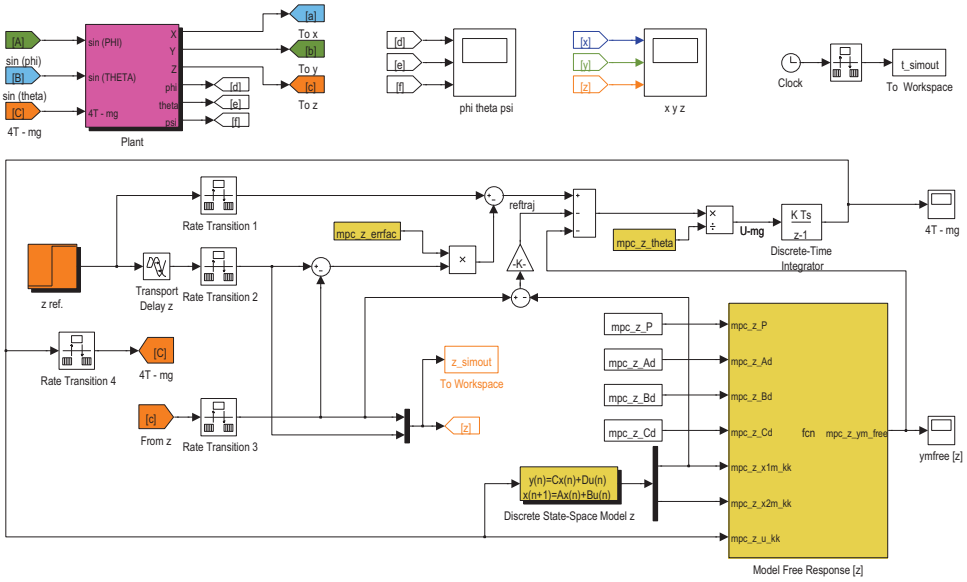

Figure 4.6: A Snapshot of the Overall MPC Control System Design – Altitude-hold Controller

contrast to the decentralized design, the new formulation does not simply add control inputs from MPC and Integral control algorithms but instead, the Integral action is incorporated in the formulation. This way, the steady state error is eliminated and the controller will

be capable of constraint handling since dynamics of the integrator is included in the QP formulation. From *Phase I* onwards, focus is on the centralized design.

4.2 Phase I: Trajectory Tracking: Autonomous Flight

Hard Constraints on the Inputs - Altitude-hold Controller As mentioned previously, there are four effectors in the form of four brushless DC motors that provide the helicopter with lift as well as directional thrust. These DC motors each, receive a PWM signal changing with time within the range of 0 to 0.1 for nominal operation. However, the lowest PWM signal corresponding to zero rotational speed of motors is shifted by 0.06. This leaves a rage of 0.06 to 0.1 to vary the angular velocity of propellers spinning from zero to a maximum of 2500 rpm or so, corresponding to the specific DC motor mounted. With such a tight operating range consideration of system's hardware constraints plays a crucial role in successful conduction of autonomous flight.

Hard Constraint on the Inputs - Lateral and Longitudinal Controllers The lateral and longitudinal controllers' manipulated variables are $\sin\phi$ and $\sin\theta$, respectively. Since the model reduction technique employed is based on the assumption that these angles stay within the vicinity of zero in almost all flight maneuvers–except for some really abrupt changes of direction or orientation which is not the case–it is crucial to keep Euler angles within the tight presumed ranges, as close as possible to zero. Otherwise, the reduced model will not precisely represent the non-linear dynamics of the quadrotor helicopter, thus rendering the control system unstable or stable but with degraded performance. Once again, this operational constraint need to be mathematically formulated and then modeled in simulation to be automatically taken care of.

Hard Constraint on the Rate of Change of the Inputs In addition, smooth transition from one flight condition to the other relies on gradual changes of a control signal. This is achieved by incorporation of hard constraints on the control variable incremental variations of manipulated variables $\sin\phi$, $\sin\theta$, and $4T - mg$ associated with the lateral, longitudinal, and altitude-hold controllers, respectively.

4.2.1 Simulation Results

Prior to implementation of the developed controller onto the hardware, it is desirable to have a rough image of how the control system would perform in flight. For highly agile unmanned vehicles as is the case for an unmanned quadrotor helicopter, this pre-implementation simulation prevents possible hazards which may rise from unanticipated performance of the controller engaged. To this end, it has been tried to mathematically model the quadrotor helicopter and then formulate all the operational hardware constraints existing in the system under study.

Integrated Design Having fully modeled the helicopter including all the previously mentioned operational limitations in terms of constraints on the *Control Variable Incremental Variation* and those on the *Amplitude of the Control Variable*, simulation may start by putting together the controller with the plant, such that a single control system is formed. As suggested in the graph, the quadrotor helicopter starts taking-off the ground at 5 seconds following commencement of simulation. It takes another 5 seconds for the system to reach 0.8 meters off the ground. Once having established in this flight level at 20 seconds, the vehicle departs on a square trajectory, staying 10 seconds on each corner. In this flight test which lasts for 60 seconds, altitude has been maintained at all times, plus a smooth but not sluggish transitions among different flight conditions are observed. Fig. 4.7 illustrates systems performance.

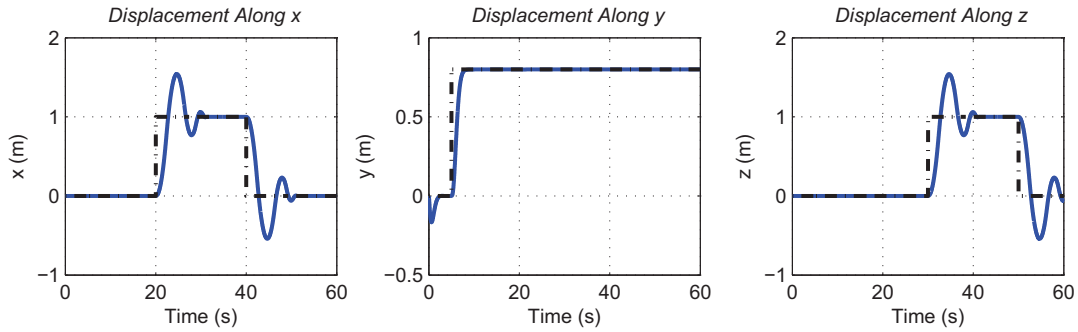

Figure 4.7: Simulation Results - 3D Tracking Performance of the MPC Autopilot Control System

4.2.2 Experimental Results

In this section, except for some fine tuning parameters such as the length of the prediction horizon N_p and that of the control horizon N_u or the penalizing parameter appearing in the cost function r_w, the simulated controller is implemented onto the Qball-X4 unmanned quadrotor helicopter available at the Networked Autonomous Vehicles Laboratory (NAVL) of Concordia University for three dimensional autonomous flight of the system. The same square trajectory has been fed into the autopilot control system as a predefined track to follow.

Generally speaking, implementation of hard constraints on the control variable incremental variation Δu is considered if there is limitation on how fast an actuator can respond to a change of the setpoint signal; which is the case in almost all practical applications. Taking one step further, this type of constraint may be implemented exclusively tighter than what the actuators manufacturer has just mentioned for their designed product. The tight treatment of constraints on rate of change of control, as long as not jeopardizing stability of the control system, makes a plant respond smoothly to the setpoint signal changes. That has been practiced in the design of this autopilot. As stated earlier and illustrated in Fig. 4.8, $\sin\phi \simeq \phi$ and $\sin\theta \simeq \theta$ are the manipulated variables of the lateral and longitudinal controllers, respectively. In order to guarantee smooth flight, they are confined to stay within a range of (-0.06, +0.06) rad as shown in Fig. 4.10 and Fig. 4.14. In contrast, this constraint has been removed for the altitude-hold controller so that not to lose agility of the system in flight level changes, as suggested by Fig. 4.12.

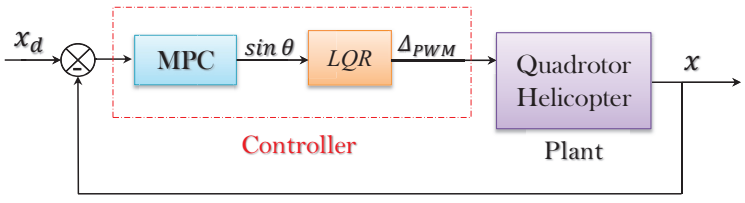

Figure 4.8: Architecture of the Longitudinal Controller

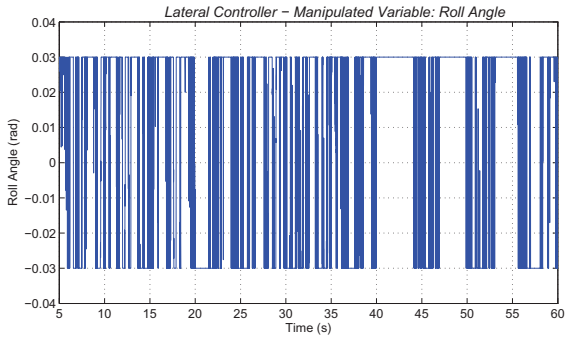

Figure 4.9: Lateral Controller – Constraints on $U : \sin\phi \sim \phi$

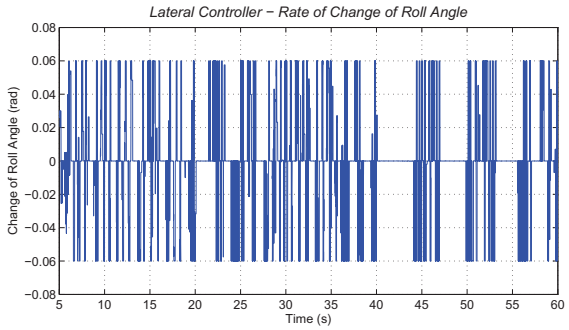

Figure 4.10: Lateral Controller – Constraints on $\Delta U : \delta \sin\phi \sim \delta\phi$

Also, implementation of hard constraints on the amplitude of control u is a must since violation of such limits means actuator saturation which is not acceptable in control. For the lateral as well as longitudinal controllers, $\sin\phi$ and $\sin\theta$ as the manipulated variables, should not deviate much from zero so as to maintain validity of linearizations and the model reduction technique used. This requirement has been met by implementation of hard constraints on them to stay within (-2, +2) degree range or (-0.03, +0.03) rad as shown in Fig. 4.9 and Fig. 4.13. For the altitude-hold controller this has been implemented as $-12 < Lift < +4$ Newton so as not to violate the acceptable rage of (0.06, 0.1) of the DC motors' PWM signal. This is suggested by Fig. 4.11.

Eventually, the offset-free tracking capability of the autopilot control system along a square trajectory for the unmanned quadrotor helicopter is illustrated in Fig. 4.15. Due to the constrained optimal control framework employed, as presented in Fig. 4.16, all the four PWM signals have stayed within their acceptable rages throughout the flight test, leaving

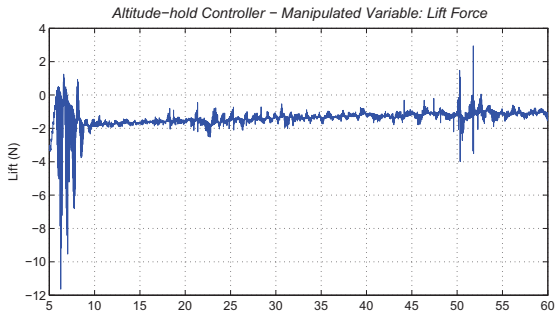

Figure 4.11: Altitude-hold Controller – Constraints on U : $(4T - mg)$

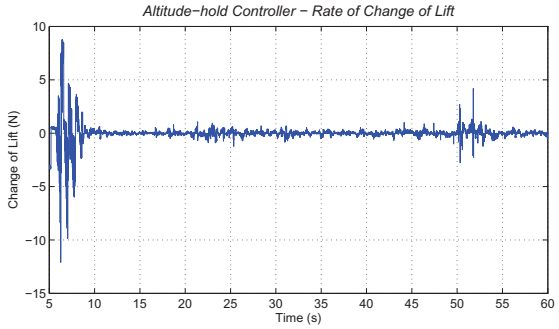

Figure 4.12: Altitude-hold Controller – Constraints on ΔU : $\delta(4T - mg)$

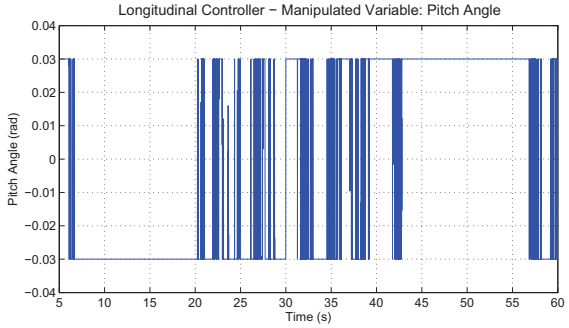

Figure 4.13: Longitudinal Controller – Constraints on U : $\sin\theta \sim \theta$

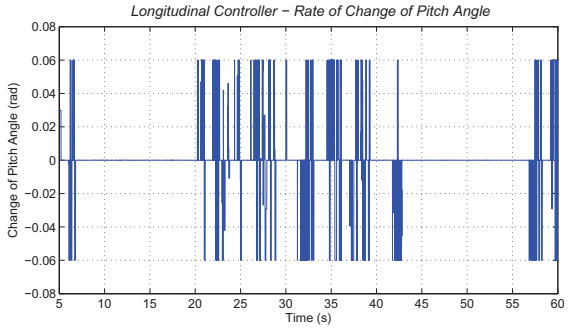

Figure 4.14: Longitudinal Controller – Constraints on ΔU : $\delta \sin\theta \sim \delta\theta$

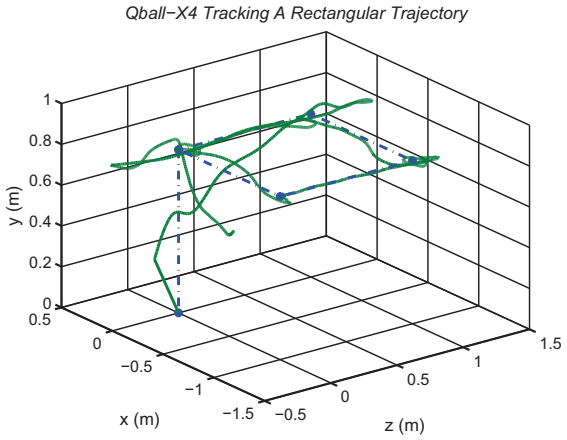

Figure 4.15: Autonomous 3D Flight along a Square Trajectory

a margin of (0.095, 1) for robustness against probable disturbances prevalent in the experimental environment.

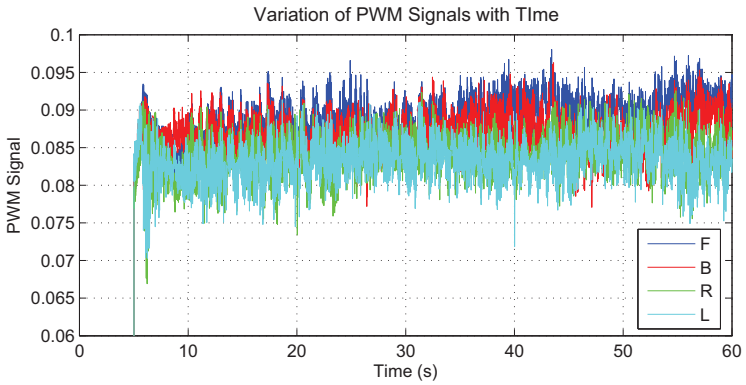

Figure 4.16: Illustration of Four PWM Signals Bounded to (0.06, 0.1)

An Important Notice Since the MPC law implements the first control movement and ignores the rest of the calculated future movements along a control signal, it is highly recommended that constraints (if there exists any) be imposed on the first control movement rather than the whole calculated future signal. Herein, this has been benefited from for the three types of the aforementioned constraints; meaning that constraints are imposed solely on the first elements in each of the $u(k)$, $\Delta u(k)$, and $y(k)$ trajectories rather than being put on all the elements. This is of great essence because for the case of Qball-X4, if

otherwise acted, real-time hardware implementation of the MPC is compromised due to limited onboard computational power.

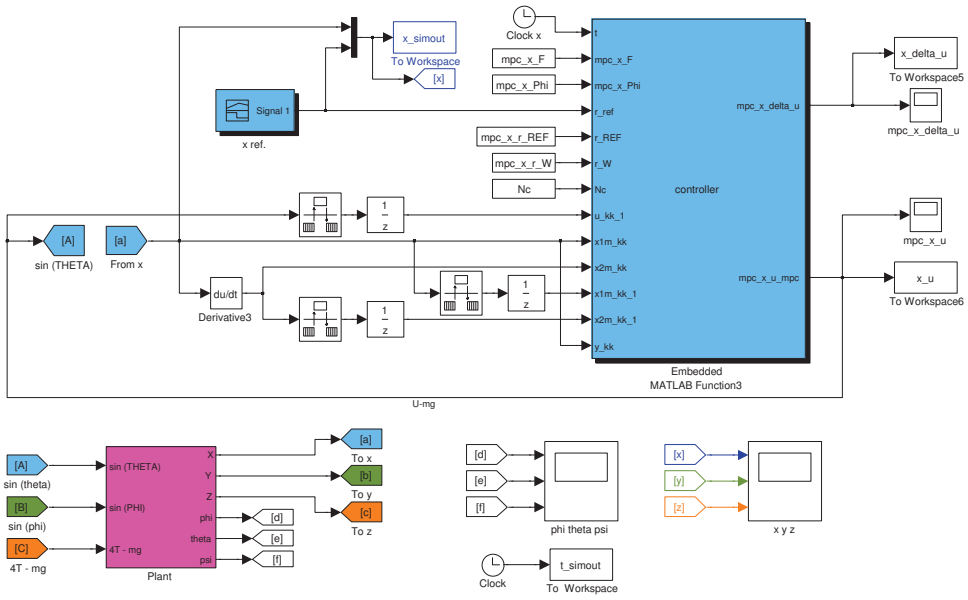

Figure 4.17: A Snapshot of the Overall MPC Control System Design – Longitudinal Controller

4.3 Phase II: Tests of Robustness to Abrupt Mass Variations

Airdrop is a very useful and common manoeuvre (technique) of flying vehicles for other different civil and military applications such as: delivery of supplies to ground forces or flight test of hypersonic and glider-type experimental airplanes which need to be mounted on another flying vehicle and to be released in the air. During the recent earthquake in Japan, military helicopters were dumping seawater on a stricken nuclear reactor in north-eastern Japan to cool overheated fuel rods inside its core. Also the U.S. Joint Forces Command continues to develop the Joint Precision Airdrop System (JPAS) with new ways of delivering supplies to ground forces while minimizing risks to soldiers. A joint military utility assessment team recently observed and rated airdrops of cargos of 6,000 to 10,000 pounds at Yuma Proving Ground, Ariz [19].

As stated previously, the problem of either linear or nonlinear control design has been addressed using several methods such as feedback linearisation [3], sliding mode control [4], and back-stepping control [5]. Nevertheless, among these studies, maximum take-off weight has always been assumed to be constant with flight time and the effects of either gradual or abrupt mass variation over the period of flight have not been well investigated. The issue of maximum take-off weight variation is of much concern since for some specific applications such as search and rescue, firefighting, and aerial spray of pesticides, to name but a few, either abrupt or gradual mass variation is inevitable.

4.3.1 Simulation Results

In this section, in order to evaluate performance of the autopilot under the effects of abrupt mass variations, the quadrotor helicopter is sent up to 70 cm off the ground and locked in place on the altitude-hold mode. A payload of 300 g is attached under the quadrotor, and will be released at some predetermined time once the system is well established at the 70 cm

flight level. In this simulation drop happens to be at 30 seconds following commencement of flight. In the first scenario, the controller allows the plant output to jump as much as it needs until it becomes stable. This is illustrated in Fig. 4.18. In the second scenario, by implementation of hard constraints on the plant output, it has been tried to reduce the jump at the instant of release. This is illustrated in Fig. 4.19.

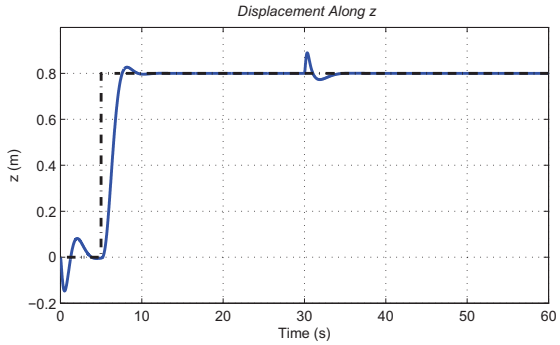

Figure 4.18: Payload Dropping under Model Predictive Control – Scenario 1

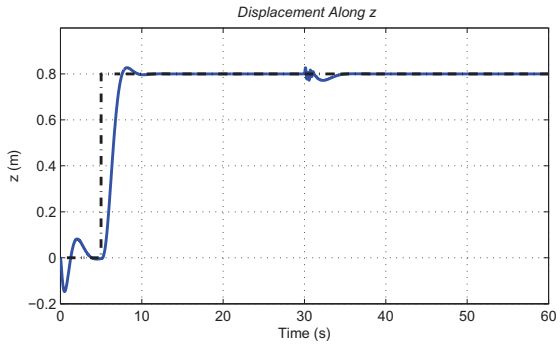

Figure 4.19: Payload Dropping under Model Predictive Control – Scenario 2

4.3.2 Experimental Results

The Payload Releasing Mechanism For the purpose of dropping a payload, a servo motor is used in a simple configuration and is installed under the quadrotor battery bay. The PWM signal generated by the Gumstix onboard computer controls position of the servo motor and emits proper commands to push/pull the metallic rod attached to the servo

horn. The payload is hooked to the metallic rod and is released upon transmission of a command at the desired time. This mechanism is shown in Fig. 4.20.

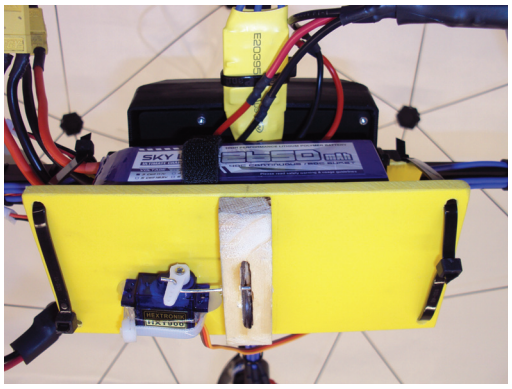

Figure 4.20: Servo based payload releasing mechanism

Generally speaking, it is intended to demonstrate how robust the control system is against probable changes of mass that might happen during the course of flight either intentionally, as is the case of a quadrotor helicopter commissioned to supply food to the victims of an earthquake or unintentionally, like fuel mass reduction that happens over the course of a long flight as fuel it consumed by the quadrotor helicopter. Autopilot's performance is depicted in Fig. 4.21.

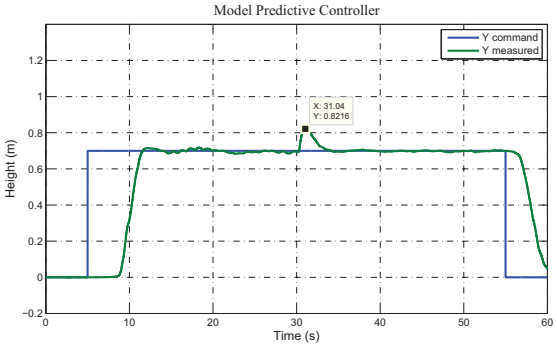

Figure 4.21: Payload Dropping under Model Predictive Control

Comparison with the Baseline Controller: Two other control techniques are studied in real time and implemented on the quadrotor UAV for performance comparison. Towards this end, two series of experiment are conducted. In the first set of experiments,

focus is on a single PID controller to take over control of the quadrotor over the phases of taking-off, hovering with payload, payload dropping, and landing. This is the baseline altitude-hold controller of the Qball-X4 with which the system comes, as a testbed for educational/research purposes. Performance of the controller is illustrated in Fig. 4.22.

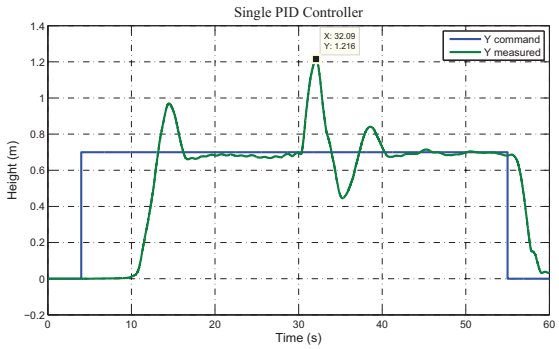

Figure 4.22: Payload Drop under a Single PID Control

Although the single PID controller is capable of keeping the desired height, it is not able to eliminate undesired overshoot at the moment of payload drop. Hence, in the second set of experiments, the single PID controller is replaced by a Gain-Scheduled PID controller to improve performance of the system at that specific moment, i.e. payload drop. This is illustrated in Fig. 4.23.

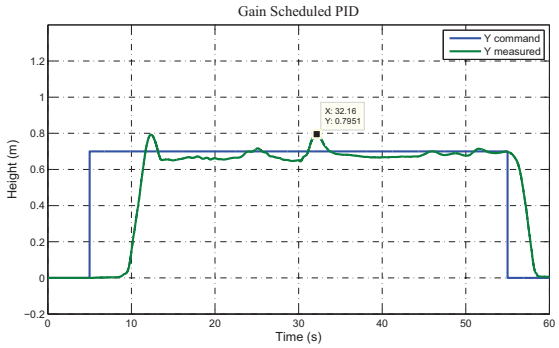

Figure 4.23: Payload Drop under a Gain-Scheduled PID Controller

For the case of a single PID controller the quadrotor does maintain the desired height but it is not satisfactory in the sense that a 73% of overshoot happens at the instant of drop. On the other hand, the GS-PID controller noticeably improves system's reaction to payload

drop, reducing vertical jerk to 13.6% of overshoot at the moment of release.

To sum it up, the MPC control technique proves to perform best, even though compared to the GS-PID controller the control system overshoots 3.4% more at the instant of payload drop. This can be decreased by proper tuning of the controller using the controller's design parameters such as the prediction horizon N_p, the control horizon N_c, and control change penalizing parameter r_w in the cost function. Under MPC, both take-off and payload carrying flight phases are better than either a single PID or a GS-PID controllers in terms of offset-free tracking and takeoff overshoot.

In terms of overshoot, GS-PID performs a better however, issues such as the tedious task of fine tuning–that may take upto a day of consecutive experiments–to find a set of proper gains, as well as dependability of successful tuning on availability of healthy and fully charged Li-Po batteries on which performance of the control system highly relies, are two essential factors that should be drawn into consideration. For instance, not fully charged battery packs can have an adverse effect on the performance of fine-tuned controller gains and deviate them from the previously found values. Furthermore, a set of finely tuned gains are effective as long as the payload's weight remains the same; meaning each a new payload is used, the whole process of finding tuned gains should be repeated. And then this question arises: How many times can a control system be exposed to such a number of repetitive experiment just for the purpose of tuning? or then is this design cost-wise or time-wise justifiable? However, theses issues are not of concern when it comes to Model Predictive Control.

As mentioned previously, in this experiment treatment of the setpoint as a constraint on the state variable or output is not practised because of limited computational power onboard the Qball-X4 quadrotor helicopter.

4.4 Phase III: Fault-tolerant Control in the Presence of Faults Induced by Reduced Actuator Effectiveness

Fault-tolerant Control "Fault-tolerant control does not yet comprise a unique theoretic framework but employs specific ideas to treat the different problems." [21] Due to the distinguishing feature of the MPC that is constraint handling, it is potentially a promising tool for fault tolerant control applications [22]–[23]. Since the MPC controller recalculates the control signal at every sampling time, in case the post-fault model is available, any change in the process model can be reflected easily into control signal computation. The constraint handling capability of MPC allows close operation to the boundaries of the tight post-fault operation envelope. The occurrence of a fault does not change a control system's objective that is expected by the prospective user. In fact, the nature of a fault-tolerant control system is to make sure that the objective(s) are met in spite of fault(s). Though there is a reach literature on fault-tolerant MPC, yet there exists little if any, on fault-tolerant MPC applied to unmanned aerial systems. This is mainly because of the mentioned strong reliance of successful-MPC-implementation on availability of high computational power onboard the unmanned airborne system. That is the main motivation for this study.

Crucial to fault-tolerant control design is providing information about the fault impact. This is the aim of the fault diagnosis unit. Development of the corresponding algorithms is out of the scope of this study; and it is assumed that the results of diagnosis are available to be used for the purpose of controller redesign or fault accommodation.

Fault Accommodation vs. Control Reconfiguration Minor faults are typically dealt with by fault accommodation where the controller's parameters are slightly modified to adapt to the system parameters of the faulty plant. In all other cases in which fault accommodation cannot be the solution, like in the case of losing a critical actuator or sensor,

control system reconfiguration is intended in which the control loop has to be reconfigured and new controller parameters are sought. Fault accommodation is distinguished from control reconfiguration according to whether the I/O signal structure between the controller and the plant is modified or not. Reconfiguration is associated with the use of a different I/O relation between the controller and the system. Switch of the system to a different internal model so as to change its mode of operation is an instance of such I/O switching. Accommodation does not employ such a means.

Active vs. Passive Fault Tolerant Control In the passive fault-tolerant control system, the control system is designed in a way that design objective is met in healthy as well as in faulty situations without any modification made to the original controller or plant. In other words, passive fault tolerant control systems could be considered as robust control systems in which the ability of achieving control system's objective is preserved, whatever the system situation, i.e. healthy or faulty. Indeed, faults can be considered as uncertainties which affect the system parameters. In contrast to passive, active fault-tolerant control is based on modification of the control law employed, so that the new law adapts to the faulty situation. Therefore, active fault-tolerant controllers implement the solution of problems solved, corresponding to either healthy or faulty situations. [24]

Generally speaking, battery-based electrical systems are prone to voltage as well as current drop after some time following commencement of current draw. Depending on the capacity of the battery pack onboard an unmanned system and energy consumption of the vehicle, the period of time for which effective use of a battery is defined before a recharge becomes required, though short or long, is limited. As time passes and the unmanned system approaches the end of its battery capacity, voltage and current drop become increasingly significant with time. This, along with other factors such as the number of loads drawing current at the same time, time-varying operation of actuators due to unpredictable

environmental effects, and other technical parameters in this regard, make it hard to provide an uninterruptable constant power supply.

Evidently, any variation in voltage or current of a power supply is reflected on operation of actuators involved, yet performance of the unmanned system. At the same time, issues such as safety, reliability, availability, and dependability of unmanned systems are required properties if such systems are one day to eliminate the need for intervention of human beings. This holds for unmanned quadrotor helicopters too. Based on a number of conducted experiments, voltage or current variation of whatever magnitude influences performance of an unmanned quadrotor helicopter; and unless the associated controller is designed with some degree of robustness to this phenomenon, the unmanned vehicle can be considered neither reliable nor available.

4.4.1 Simulation Results

In this simulation it will be tried to expose the unmanned quadrotor helicopter to the effects of voltage/current drop in order to evaluate performance of the designed autopilot in terms of being fault-tolerant to variations in voltage/current drop. This has been implemented by 10% collective reduction in actuator effectiveness. As for the case of the healthy system, a square trajectory is defined to be tracked. In addition, it is assumed that there is no diagnosis unit providing information regarding fault detection, isolation, and identification, even though this information is manually fed into the control system by an operator. In the first scenario, the controller allows the plant output to jump as much as it needs until it becomes stable. This is illustrated in Fig. 4.24. In the second scenario, by implementation of hard constraints on the plant output, it has been tried to reduce the jump at the instant of release. This is illustrated in Fig. 4.25.

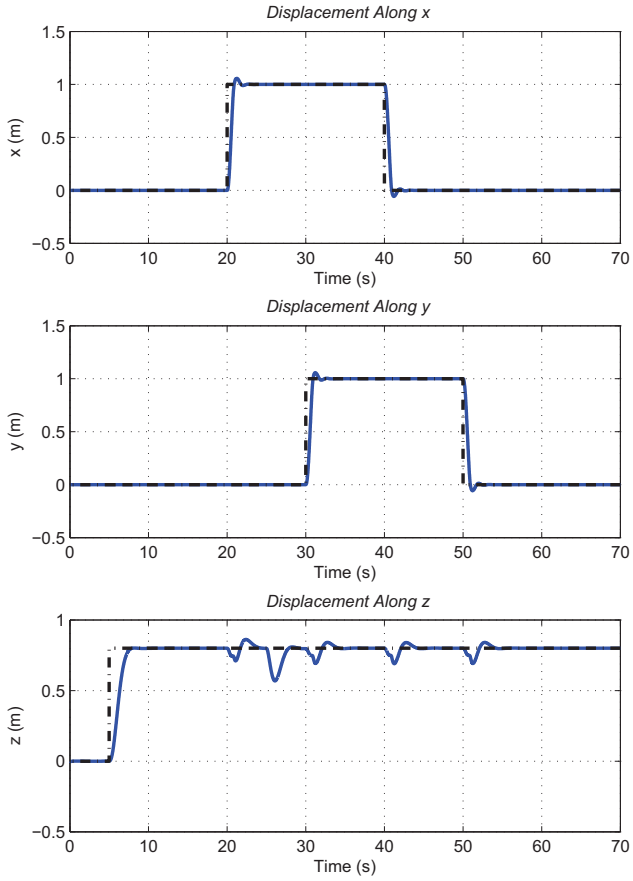

Figure 4.24: 10% Collective Reduction in Actuator Effectiveness at $t = 25s$– Scenario 1

4.4.2 Experimental Results

In this section, except for some finely tuned parameters which will be slightly changed, the very same simulated controller is implemented onto the ball-X4 unmanned quadrotor helicopter without any major modification. For three-dimensional autonomous flight of the unmanned quadrotor, the same square trajectory has been fed into the control system as a predefined track to follow.

Collective Reduction in Actuator Effectiveness As illustrated in Fig. 4.26–4.29, the fault is injected artificially at $t = 25s$ following take-off when the quadrotor is on the verge of departing on a turn at the second corner. This is one of the most critical phases of flight

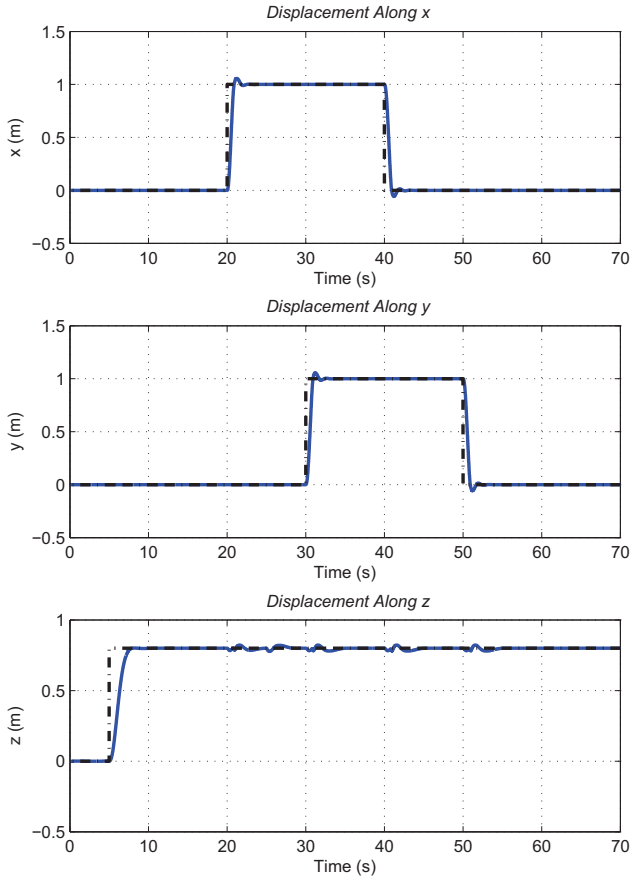

Figure 4.25: 10% Collective Reduction in Actuator Effectiveness at $t = 25s$ – Scenario 2

because lateral and longitudinal controllers are actively engaged to make a right-angle turn. If the controller is fault-tolerant enough against reduction in actuator effectiveness while the system is about to depart on a turn, then satisfactory performance is guaranteed if the fault happens at other regions of the flight envelope as well. Regardless of the altitude of the quadrotor helicopter at which fault occurs, the vehicle retains altitude at the cost of 70 cm temporary loss of height over a couple of seconds; then re-establishes itself at the same flight level as before. Though 70 cm loss of altitude might be noticed at the flight level of 80 cm–as studied here–it is almost negligible while the unmanned helicopter is in operation in its nominal mission such as firefighting or surveillance during which the system is flying at flight levels of 3000 to 6000 ft. As suggested by Fig. 4.26–4.29 the developed MPC

framework preserves the control system's ability to meet the trajectory tracking objective envisioned for that, in healthy conditions as well as faulty situations, rendering the control system *passive fault-tolerant*.

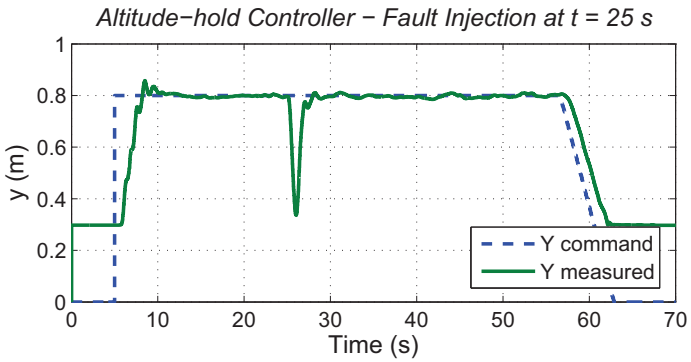

Figure 4.26: Four Faulty DC Motors - Altitude-hold Controller

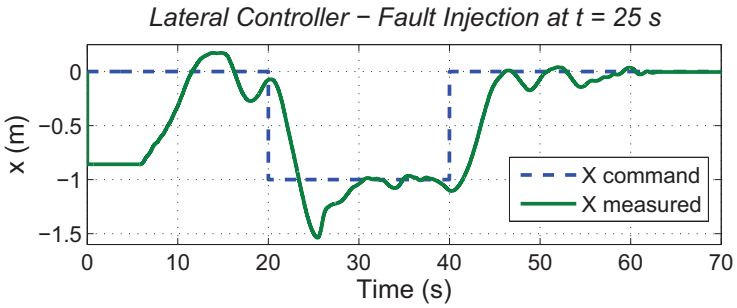

Figure 4.27: Four Faulty DC Motors - Lateral Controller

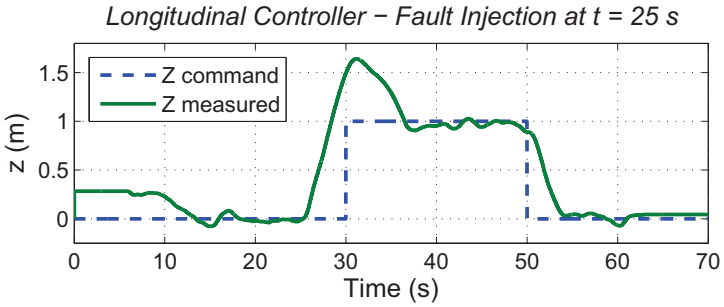

Figure 4.28: Four Faulty DC Motors - Longitudinal Controller

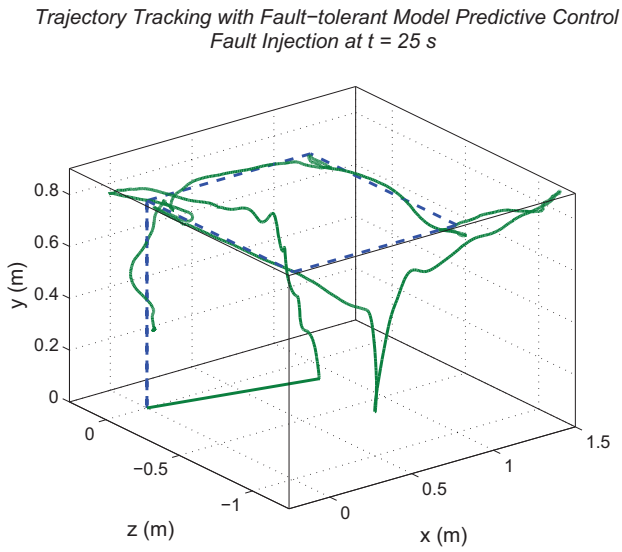

Figure 4.29: Four Faulty DC Motors - Trajectory Tracking

Comparison with the Baseline Controller: Compared with that of the Qball-X4's baseline controllers in which a combination of LQR and PID techniques are used, employment of the MPC can essentially improved system's behaviour in terms of reliability upon occurrence of collective reduction in actuator effectiveness. As illustrated in Fig. 4.30, the baseline controller is not capable of handling a 10% collective reduction in all the four DC motors simultaneously and shows distress by touching the ground for a some seconds. This is in contrast with the satisfactory results obtained from the experiments conducted for the same amount of collective reduction, but under the MPC technique. Three dimensional tracking performance of the baseline controller is depicted in Fig. 4.31.

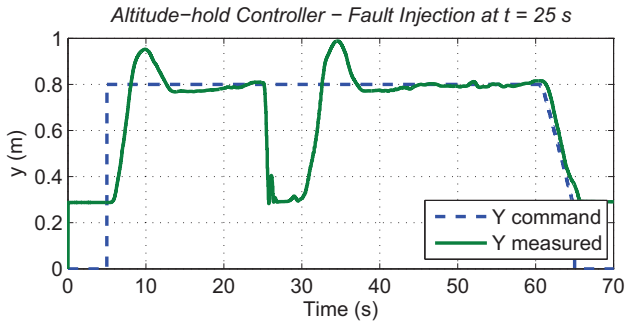

Figure 4.30: Four Faulty DC Motors - Baseline Altitude-hold Controller

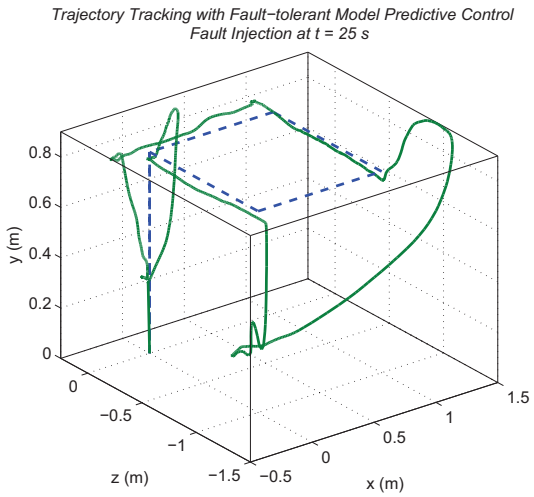

Figure 4.31: Four Faulty DC Motors - Trajectory Tracking with the Baseline Controller

Singular Reduction in Actuator Effectiveness Multi-thruster aerial vehicle in which thrusters operate in parallel to provide the control system with sufficient lift or thrust are prone to a second fault as well; that is unbalanced/asymmetric loss of actuator or thruster effectiveness. This is of great concern because malfunction of a single actuator rather than all, gives rise to development of unbalanced/asymmetric forces and moment which will consequently lead to instability of the vehicle, preventing the system from mission fulfilment. Singular loss usually is not a consequent of voltage or current drop but mainly evolves from faults occurring among onboard electronic boards or burned electronic elements. In this experiment, the quadrotor helicopter is exposed to 10% single actuator loss of effectiveness at $t = 35s$. Herein, even though the reconfiguration mechanism does not rely on the information regarding fault isolation, it does require processed data returned by the diagnosis algorithm concerning fault detection and identification. Therefore, it is assumed that there is a diagnosis unit providing precise information regarding fault detection and identification but not isolation.

Once occurrence of a fault is detected and its magnitude is identified, the reconfiguration mechanism relaxes operational constraints put on inputs' rate of change ΔU accordingly, bringing agility back to the system of quadrotor so that the helicopter can

compensate for the happened deviations from the predefined trajectory as fast as possible, avoiding growth of the fault to instability or consequent failure of the whole system. That is achieved at the expense of minor instability introduced into the system which manifests itself as lack of smoothness in motion while the trajectory is being tracked by the quadrotor helicopter. This degraded performance is completely acceptable in order to refrain from failure. This is illustrated in Fig. 4.32–4.35.

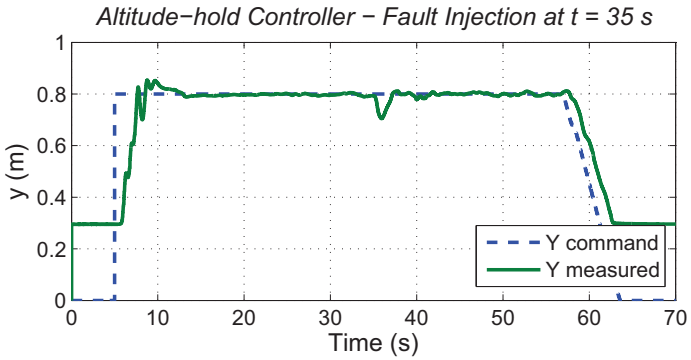

Figure 4.32: One Faulty DC Motor - Altitude-hold Controller

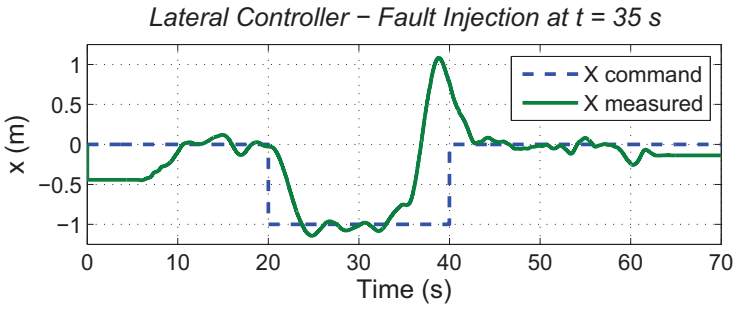

Figure 4.33: One Faulty DC Motor - Lateral Controller

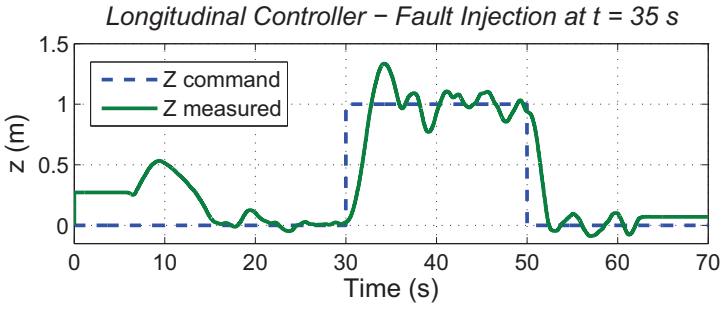

Figure 4.34: One Faulty DC Motor - Longitudinal Controller

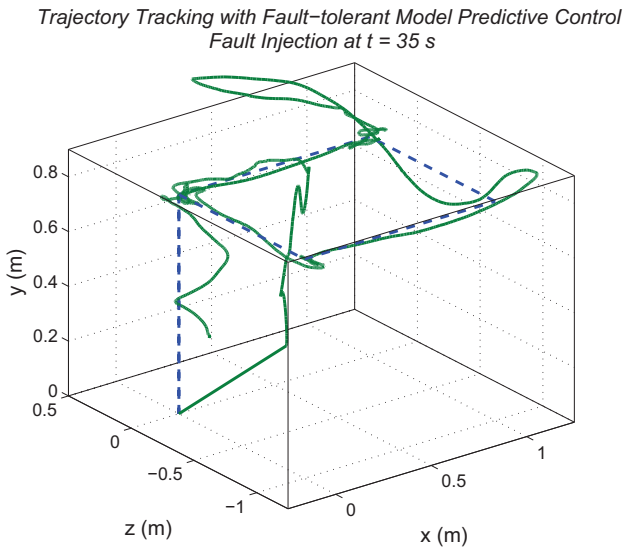

Figure 4.35: One Faulty DC Motor - Trajectory Tracking

Comparison with the Baseline Controller: Once again, compared with that of the Qball-X4's baseline controllers in which a combination of LQR and PID techniques are used, employment of the MPC has essentially improved system's behaviour in terms of reliability upon occurrence of singular reduction in actuator effectiveness. As illustrated in Fig. 4.36, even though the baseline controller is capable of handling a 10% singular reduction in one of the four DC motors, loss of height is significant and cannot be reduced further. The loss of heigh is obviously less in the experiments conducted for the same amount of singular reduction, but under the MPC technique. Three dimensional tracking performance of the baseline controller is depicted in Fig. 4.37.

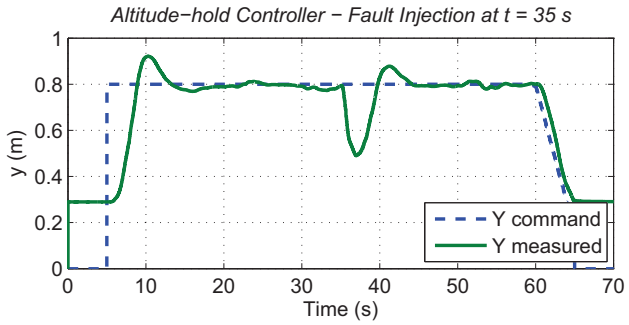

Figure 4.36: One Faulty DC Motor - Baseline Altitude-hold Controller

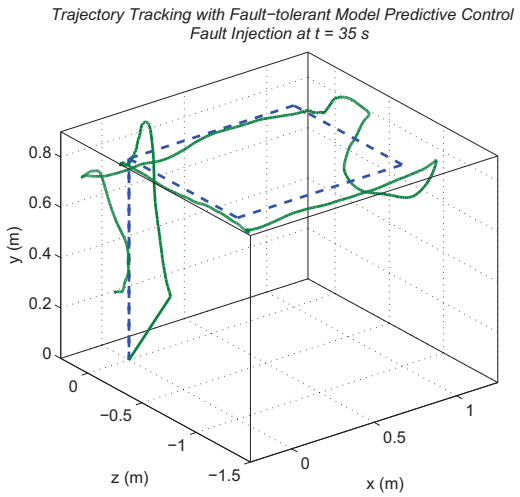

Figure 4.37: One Faulty DC Motor - Trajectory Tracking with the Baseline Controller

Under theses two most probable faulty scenarios, namely collective reduction of actuator effectiveness due to voltage or current drop, and singular loss of effectiveness as a result of fault occurrences on the onboard electronic boards, the controller proved to be effective in simulation and implementation on the unmanned quadrotor helicopter. Over the course of this study, existence of a diagnosis unit providing precise information regarding fault detection, isolation, and identification has been assumed. Even though in both scenarios fault accommodation and control reconfiguration introduce a limited extent of lost smoothness in motion while trajectory tracking, this amount of degradation of performance is regarded as acceptable since it prevents a bigger event from taking place, that is failure of the whole system.

Summary In this section, the efficient formulation was implemented in practice for the sole purpose of tracking a rectangular trajectory to further illustrate reliance of the efficient formulation on availability of a concise mathematical model of the plant. This was followed by three stages of study and development towards the end of:

- Three dimensional autonomous tracking within the framework of constrained MPC;
- Proof of robustness against gradual/abrupt mass variations, both in simulation and

practice; and

- Proof of being fault-tolerant against singular/collective reduction in actuator effectiveness.

The following section concludes the thesis and suggests a direction for the future work of this study.

Chapter 5

Conclusion

Since introduction of Model Predictive Control, its use has centred around process control. As mentioned earlier, this is mainly due to the facet that implementation of MPC strongly relies on availability of high computational power because of repetitive nature of computations involved. On the other hand, the requirement of high computational power implies presence of a fairly large system, like a general purpose personal computer to accommodate the MPC controller along with its iterative time-consuming computations. This is not possible for unmanned aerial system; because essentially there is neither enough space nor payload capacity onboard the vehicle to take in a huge amount of circuitry and electronic boards. Therefore, there are two options; either attention should be turned to other control techniques other than MPC, or effort should be made to reduce the burden of calculations such that a light weight single-board computer or microcontroller can handle all the calculations corresponding to MPC. In this work, focus has been on the latter. Model reduction techniques which basically reduce complexity of a plant model yet still preserving dynamics of the plant, can greatly contribute to reduction in computational loads; this has been practiced in the development of the designed autopilot control system. In addition, various fast optimization algorithms have evolved over years. They can be suitably used to efficiently solve a quadratic programming problem, as is the case for a constrained MPC.

In this study, a fast QP solver known as Hildreth's Quadratic Programming Procedure has been made use of to solve the iterative optimization problem involved in the implementation of MPC; this has proved to be a success.

In addition to availability of high computational power, success of the MPC implementation is tightly dependent on existence of a precise mathematical model of the plant. Even though derivation of such an elaborate model is mathematically doable, in practice there always exists some degree of discrepancy between the mathematical model and the real plant, thus the requirement of offset-free tracking is hardly attainable within the framework of MPC, unless measures are taken to resolve this issue. From the classical control theory proper employment of the integral action control can eliminate the steady state error rising from such model mismatch and discrepancies prevalent in control systems. As discussed, this gives rise to two control system design approaches, centralized design versus decentralized design. In a centralized design, in contrast to the decentralized design, the control inputs from MPC and Integral algorithms are not simply added up but instead, the Integral action is incorporated in the MPC formulation. This way, the steady state error can be eliminated and the control structure preserves its capability of constraint handling since the integrator dynamics is included in the QP formulation. In this study the two approaches have been investigated, both in simulation and practice.

Eventually, with the aid of concepts and techniques already stated, a three dimensional autopilot control system within the framework of MPC is developed and tested through numerous flight tests conducted in the Networked Autonomous Vehicles Laboratory of Concordia University. The overall performance of the quadrotor helicopter is evaluated under autonomous fight for three scenarios of *Trajectory Tracking*, *Payload Drop Mission*, and *Robustness to Voltage and Current Drop*. Both simulation and experimental results just presented demonstrate success of the design.

Future Work Suggestions for the future extension of this work include:

- Further investigation on availability of possibly more efficient optimization algorithms, specifically those regarding the known quadratic programming problem as appeared in the MPC;
- Further investigation on better employment of various model reduction techniques in order to preserve as much dynamics as possible of the plant, yet achieving greater accuracy;
- Use of less accurate onboard and outboard sensors–instead of the precise Optitrack–for date measurement and feedback, in order to determine vulnerability of the MPC design to precision of current state measurements; and
- Use of the same developed autopilot control system for a fixed wing testbed in the form of a *Wing Leveller*. Lateral equations of motion of an airplane have their own approximate modes. One of such modes corresponds to the pure rolling motion. This approximate mode is effectively used in the design of wing leveller autopilots [26]. Since it is a first order transfer function of a SISO system, it can be a good starting step in transition from rotary wing to fixed wing aircraft.

Appendix A

LQR Controller Design

A.1 LQR (Linear Quadratic Regulator)

Optimal Control is an area within the theory of control that deals with control of dynamic systems in a way that one specific designer-defined function is minimized. This specific, designer-defined function is also known as *Cost Function*. Specifically speaking, the case in which the dynamics of the system is governed by a set of linear differential equations and the cost function is described by a quadratic function, is called *Linear Quadratic problem* (LQ Problem); the answer to this problem is LQR or Linear Quadratic Regulator which is basically a full state feedback controller.

A.1.1 Design of a Regulator

LQR State Feedback Design Assuming a control system expressed in state space format as:

$$\dot{x} = Ax + Bu \tag{A.1}$$

where x is the *State Vector*, A is the *State Matrix*, B is the *Control Matrix*, and u is the *Control Vector*. Provided that all the states are available for measurement, a State Variable

Feedback Controller is designed as:

$$u = -Kx + v \tag{A.2}$$

By substituting (A.2) in (A.1), the state space representation of the closed loop system becomes:

$$\dot{x} = (A - BK)x + Bv = A_{NEW}x + Bv \tag{A.3}$$

This way, the closed loop propertied of the system can be determined by proper assignment of poles of the system. As can be seen the output matrix does not play a role in state feedback controller design.

As control systems grow in terms of complexity, it is no longer possible to make use of pole placement techniques in order to determine the location of the closed loop system poles. In other words, for such systems the known Achermann's formula is inconvenient for determination of all closed loops of the system. That is the reason why attention is turned to a method which is capable of addressing this problem, no matter what the order of the system is. To this end, a cost function is required to be defined as the performance index, which should be minimized in one way or another. Then, the solution to this minimization problem is the optimized gain that has been sought as the gain of the intended state feedback controller. This performance index is defined by:

$$J = \frac{1}{2}\int_0^\infty (x^T Qx + u^T Ru)dt \tag{A.4}$$

As this equation suggests there are two design parameters Q and R that should be decided on prior to design. Q should be chosen to be positive semi-definite, while R needs to be positive definite. They are Weighting Factors with significance. Keeping the value of Performance Index the same, as the value of Q increases, x will decrease and vice versa. The same comes true for R. In other words, a big value of Q will keep the error signal small,

whereas a big value of R will keep the control signal quite small. Based on the criteria and requirements, it is the job of an experienced control engineer to decide on the relative value of these two design parameters. Needless to say, different values of weighting matrices cause the system exhibit different transient and steady state performance.

For the time being, it is assumed that the input v is equal to zero,thus the only concern is stability of the system rather than following a specific reference input. That is the reason why it is named "Regulator". As the name implies it brings all state variables to zero and stabilizes the control system. In this type of controller the system does not accept a command signal, contrary to the tracking problems. Compared with other controllers, one of the traits that set LQR apart is possibly the robustness presented by this control technique.

Considering MATLAB as the controller design tool, development process of a Regulator simply accounts for expression of the system dynamics in state space format (Matrices A, B, C, and D) plus determination of weighting matrices Q and R. As the values of these two design parameters directly affect system performance, decision making in this phase of design should be with care. Q and R can be both identity matrices. In this case all elements of the error signal (or the input signal) are treated equally the same, thus none of them has superiority over others. Next, the solution of the previously talked about optimization problem is found by entering the command $K = lqr(A,B,Q,R)$. It should be notified that this optimization problem is guaranteed to produce a feedback gain vector stabilizing the control system as long as the studied system is *observable*.

A.1.2 Design of a Setpoint Tracker

LQR State Feedback Design Having assumed a control system expressed in the state space format:

$$\dot{x} = Ax + Bu \tag{A.5}$$

The same control signal as that of the *Regulator* design is constructed via the feedback control law:

$$u = -K\tilde{x} \tag{A.6}$$

However, contrary to the non-tracker controller, the $\tilde{x}$ vector is not simply the state vector x of the control system. In fact, it contains both the state vector and integrals of the error signal. That is to say:

$$\tilde{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_n & z_{d1} & z_{d2} & \dots & z_{dm} \end{bmatrix} \tag{A.7}$$

in which $x = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}$ indicates the state vector containing n state variables, and $z_d = \begin{bmatrix} z_{d1} & z_{d2} & \dots & z_{dm} \end{bmatrix}$ where:

$$z_{di} = \int (x_i - x_{di})\, dt\,; \quad i = 1, 2, \dots, m \tag{A.8}$$

With this definition the new representation of the system becomes:

$$\dot{\tilde{x}} = \begin{bmatrix} A & \begin{bmatrix} 0 \end{bmatrix}_{n\times m} \\ \begin{bmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix}_{m\times n} & \begin{bmatrix} 0 \end{bmatrix}_{m\times m} \end{bmatrix} \tilde{x} + \begin{bmatrix} B \\ \begin{bmatrix} 0 \end{bmatrix}_{m\times 1} \end{bmatrix} u + \begin{bmatrix} \begin{bmatrix} 0 \end{bmatrix}_{n\times m} \\ \begin{bmatrix} -1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & -1 \end{bmatrix}_{m\times m} \end{bmatrix} \begin{bmatrix} x_{d1} \\ x_{d2} \\ \ddots \\ x_{dm} \end{bmatrix} \tag{A.9}$$

Or in a more compact form the augmented control system is:

$$\dot{\tilde{x}} = \tilde{A}\tilde{x} + \tilde{B}u + \tilde{B}_d d \tag{A.10}$$

Once the state space representation of the augmented control system is obtained, design of the tracker LQR controller follows the same procedure as that of the non-tracker (Regulator). Simply, based on the desired system performance (both transient and steady state), the values are matrices Q and R are selected by the control engineer. Next, the solution of the previously talked about optimization problem is found by entering the command $K = lqr(\tilde{A}, \tilde{B}, Q, R)$. A snapshot of the overall LQR control system design for the unmanned quadrotor helicopter is presented in Fig. A.1.2.

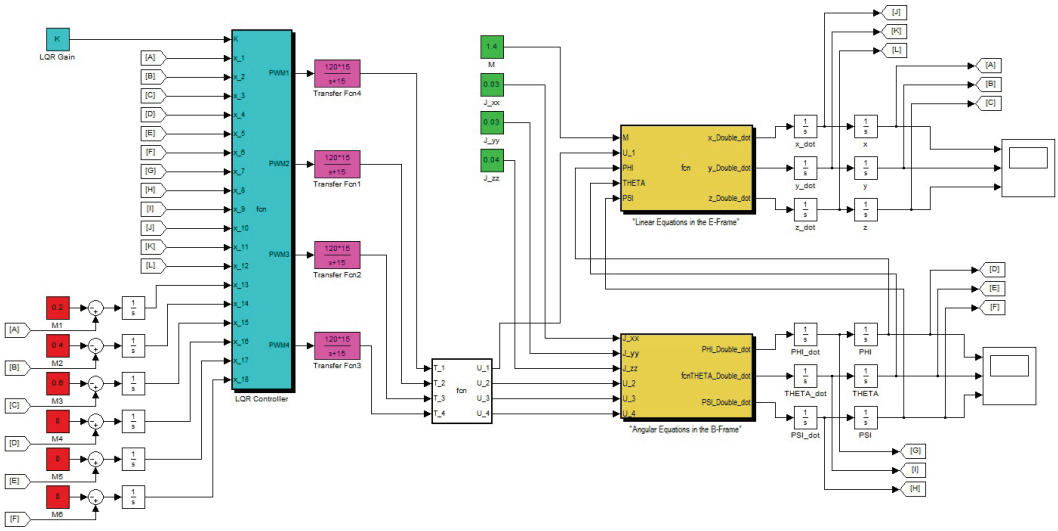

Figure A.1: A Snapshot of the Overall LQR Control System Design

Appendix B

Constrained Optimization

B.1 Quadratic Programming

Like linear programming problems, another optimization problem which can be solved in a finite number of steps is a *Quadratic Programming* (QP) problem. This is a problem in which the objective function $q(x)$ is quadratic and the constraint functions $c_i(x)$ are linear. The problem is to find a solution $\breve{x}$ to minimize:

$$q(x) = \frac{1}{2}x^T Gx + g^T x \tag{B.1}$$

subject to:

$$A^T x = b \tag{B.2}$$

$$M^T x \geq \gamma \tag{B.3}$$

This problem may be infeasible or the solution may be unbounded; however these possibilities are easily detected in the algorithms, so for the most part it is assumed that a solution $\breve{x}$ exists. IF the Hessian matrix G is positive semi-definite, $\breve{x}$ is a global solution, and if G is positive definite, $\breve{x}$ is also unique. These results follow from the convexity of $q(x)$. When

the Hessian G is indefinite then the local solutions which are not global can occur.

B.1.1 Equality Constraints

Elimination of Variables

This section studies how to minimize the objective function subject to equality constraints, i.e.:

$$q(x) = \frac{1}{2}x^T Gx + g^T x \tag{B.4}$$

subject to:

$$A^T x = b$$

It is assumed that there are $m \leq n$ equality constraints where A is $n \times m$ collecting column vectors a_i, and x is $n \times 1$. Also, it is assumed that A has rank m; if the constraints are consistent this can always be achieved by removing dependent constraints, though there may be numerical difficulties in recognizing this situation.

A straightforward way of solving B.4 is to use the set of constraints to eliminate all variables except for one. By introducing partitions to each of the matrices involved:

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}, \quad g = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}, \quad G = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \tag{B.5}$$

where x_1 is of the size $m \times m$ and x_2 of the size $(n-m) \times (n-m)$, then B.5 becomes ${A_1}^T x_1 + {A_2}^T x_2 = b$, and is readily solved to give x_1 in terms of x_2:

$$x_1 = {A_1}^{-T}(b - {A_2}^T x_2) \tag{B.6}$$

Substituting this into $q(x)$ introduces a new problem: minimize $\tilde{q}(x_2)$ where $\tilde{q}(x_2)$ is the

quadratic function:

$$\begin{aligned}\tilde{q}(x_2) = \frac{1}{2}{x_2}^T(G_{22} - G_{21}{A_1}^{-T}{A_2}^T - A_2{A_1}^{-1}G_{12} + A_2{A_1}^{-1}G_{11}{A_1}^{-T}{A_2}^T)x_2 \\ + {x_2}^T(G_{21} - A_2{A_1}^{-1}G_{11}){A_1}^{-T}b + \frac{1}{2}b^T{A_1}^{-1}G_{11}{A_1}^{-T}b \\ + {x_2}^T(g_2 - A_2{A_1}^{-1}g_1) + {g_1}^T{A_1}^{-T}b \end{aligned} \tag{B.7}$$

A unique minimizer $\breve{x}_2$ exists if the Hessian $\nabla^2\tilde{q}$ in the quadratic term is positive definite. In this case $\breve{x}_2$ is obtained by solving the linear system $\nabla^2\tilde{q}(x_2) = 0$; then $\breve{x}_1$ is found by substitution of $\breve{x}_2$ in B.6.

Lagrangian Method

In order to minimize the objective function B.4 subject to equality constraints B.5 the method of *Lagrange multipliers* introduces an alternative way of deriving the solution $\breve{x}$ and the associated multipliers $\breve{\lambda}$. To this end, the *Lagrangian function* is defined as:

$$L(x,\lambda) = \frac{1}{2}x^TGx + g^Tx - \lambda^T(A^Tx - b) \tag{B.8}$$

It is evident that the value of B.8 subject to the equality constraints B.5 is the same as the original objective function. Therefore, now B.8 is considered as the new objective function in $n+m$ variables x and λ, where n is the dimensions of x and m is the dimensions of λ. The procedure of minimization is to take the first partial derivative with respect to the vectors x and λ, and then set these expressions to zero. That is to say:

$$\begin{aligned} \nabla_x L = 0: \quad & Gx + g - A\lambda = 0 \\ \nabla_\lambda L = 0: \quad & A^Tx - b = 0 \end{aligned}$$

which can be arranged to give the linear system:

$$\begin{bmatrix} G & -A \\ -A^T & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = - \begin{bmatrix} g \\ b \end{bmatrix} \tag{B.9}$$

The coefficient matrix is referred to as the *Lagrangian matrix* and is symmetric but not positive definite. If the inverse exists and is expressed as:

$$\begin{bmatrix} G & -A \\ -A^T & 0 \end{bmatrix}^{-1} = \begin{bmatrix} H & -T \\ -T^T & u \end{bmatrix} \tag{B.10}$$

then the solution to B.9 is:

$$\breve{x} = -Hg + Tb \tag{B.11}$$

$$\breve{\lambda} = T^T g - Ub \tag{B.12}$$

These relationships were used by Fletcher (1971)to solve the equality constraint problem that evolves in the active set method [17]. This will be explained in the following sections. Explicit expressions for H, T, and U when G^{-1} exists are:

$$H = G^{-1} - G^{-1}A(A^T G^{-1} A)^{-1} A^T G^{-1}$$

$$T = G^{-1}A(A^T G^{-1} A)^{-1}$$

$$U = -(A^T G^{-1} A)^{-1}$$

B.1.2 Inequality Constraints

Active Set Methods

Most QP problems involve inequality constraints and so can be expressed in the form given in B.3. This section describes how methods for solving equality constraints can be employed to handle the inequality problem by means of an *active set* method. Most common is the *primal active set* method. This is described in the case that the Hessian matrix G is positive definite which ensures that any solution is a unique global minimizer, and that some potential difficulties are avoided. Later in this section the possibility of a *dual active set* method is considered, although this is only applicable to the case that G is positive Definite.

Primal Active Set Method In the primal active set method certain constraints, indexed by the *active set* Ξ, are regarded as equalities whilst the rest are temporarily disregarded, and the method adjusts this set in order to identify the correct active constraints at the solution to B.1 subject to B.3. On iteration k a feasible point $x(k)$ is known. This satisfies the active constraints as equalities. Each iteration attempts to locate the solution to an equality problem in which only the active constraints occur. This is most conveniently done by shifting the origin to $x(k)$ and looking for a correction $\delta(k)$. Since the newly evolved problem is an equality problem, any of the methods previously mentioned for optimization problems with equality constrains can well serve the purpose. If there exists a feasible $\delta(k)$, then the next iterate is taken as $x(k+1) = x(k) + \delta(k)$. If not, then a line search is made in the direction of $s(k)$ and choosing the step $\alpha(k)$ to find the best feasible point, therefore $x(k+1) = x(k) + \alpha(k)s(k)$. If $\alpha k < 1$, then a new constraint becomes active and is added to the active set Ξ. To put it in a nutshell, at each step of the active set method, an equality optimization problem is solved. If all the Lagrange multipliers are non-negative, then the point is a local solution to the original problem, if on the other hand there exists a $\lambda_i < 0$, then the objective function can be further reduced by relaxing the corresponding constraint

i. This is done by deleting it from the set of active constraints Ξ.

As the process goes on, it is important to monitor the value of other constraints not involved in the equality problem to make sure that they are not violated. It often happens that while moving on the working surface, a new constraint boundary is encountered. It is necessary to add this constraint to the working set, then continue with the redefined working surface.

Dual Active Set Method The family of active set methods belongs to the group of primal methods. In this group of problems, it is required that the active constraints be identified along with the optimal decision variable. However, if there are a number of constraints involved, the computational load is pretty large. In addition, the programming of this optimization method is not straightforward.

The dual active set method can be used alternatively to systematically identify the constraints that are not active in the current iteration. In this new formulation the Lagrange multipliers change names to dual variables.

The dual problem to the original primal problem is derived as follows. Assuming that there exists an x such that $M^T x > \gamma$, the primal problem is equivalent to:

$$max_{\lambda} \quad min_x[\frac{1}{2}x^T Gx + g^T x + \lambda^T (Mx - \gamma)]; \quad \lambda \geq 0 \tag{B.13}$$

Therefore, the minimization over x is unconstrained and is calculated by:

$$x = -G^{-1}(g + M^T \lambda) \tag{B.14}$$

By substituting this into B.13, the dual problem is written as:

$$max[-\frac{1}{2}\lambda^T H\lambda - \lambda^T K - \frac{1}{2}g^T G^{-1} g]; \quad \lambda \geq 0 \tag{B.15}$$

where the matrices H and K are defined as:

$$H = MG^{-1}M^T$$

$$K = \gamma + MG^{-1}g$$

Therefore, the dual problem is also a quadratic programming problem with γ as its decision variable. Equation B.15 is equivalent to:

$$min[\frac{1}{2}\lambda^T H\lambda + \lambda^T K + \frac{1}{2}\gamma^T G^{-1}\gamma]; \quad \lambda \geq 0 \tag{B.16}$$

It should be noted that the dual problem may be much easier to solve compared with the primal problem because the constraints are simpler.

To sum it up, the set of optimal Lagrange multipliers that minimize the dual objective function $J(\lambda)$:

$$J(\lambda) = \frac{1}{2}\lambda^T H\lambda + \lambda^T K + \frac{1}{2}\gamma^T G^{-1}\gamma \tag{B.17}$$

subject to $\lambda \geq 0$, are denoted as λ_{act}, and the corresponding constraints are described by M_{act} and γ_{act}. It can be easily demonstrated that with the values of M_{act} and γ_{act}, the primal variable vector x is obtained by:

$$x = -G^{-1}g - G^{-1}M_{act}^T\gamma_{act} \tag{B.18}$$

In this group of problems, i.e. dual problems, the solutions are based on auxiliary variables that are collectively referred to as dual variables λ_i, whereas in the primal problems the solutions are based on the decision variables that are collectively referred to as primal variables $\breve{x}_i$ [12].

A simple algorithm, called Hildreth's quadratic programming procedure was proposed by Luenberger (1969) for solving the dual problem. In this algorithm, the direction

vectors are selected to be equal to the basis vectors $e_i = \begin{bmatrix} 0 & 0 & \dots & \dots & 0 & 0 \end{bmatrix}^T$. Then the λ vector can be varied one component at a time. At a given step in the process, having obtained a vector $\lambda \geq 0$, attention is fixed on a single component λ_i. λ_i is adjusted to minimize the objective function. If this requires $\lambda_i < 0$, it is set as $\lambda_i = 0$. In either case, the objective function is decreased. Then the next component λ_{i+1} is considered. If one complete cycle through the components is considered to be one iteration taking the vector λ_i to λ_{i+1}, the method can be expressed explicitly as:

$$\lambda_i{}^{m+1} = max(0, \omega_i{}^{m+1}) \tag{B.19}$$

with

$$\omega_i{}^{m+1} = -\frac{1}{h_{ii}}[k_i + \sum_{j=1}^{i-1} h_{ij}\lambda_i{}^{m+1} + \sum_{j=i+1}^{n} h_{ij}\lambda_i{}^{m}] \tag{B.20}$$

where the scaler h_{ij} is the ij^{th} element in the matrix H and k_i is the i^{th} element in the vector K. The converged $\breve{\lambda}$ vector contains either zero or positive values of the Lagrange multipliers, therefore the optimal solution to the primal problem is:

$$x = -G^{-1}(g + M^T\breve{\lambda}) \tag{B.21}$$

It should be noted that the Hildreth's quadratic programming algorithm is an element-by-element search, therefore it does not require any matrix inversion, thus it always converges. This is crucial for a reliable control framework [12].

Bibliography

[1] Oner K., Cetinsoy E., Unel M., Aksit M., Kandemir I., Gulez K.: Dynamic Model and Control of a New Quadrotor UAV with Tilt-wing Mechanism. In: World Academy of Science, Engineering and Technology (2008)

[2] Erginer B., Altug, E.: Modeling and PD Control of a Quadrotor VTOL Vehicle. In: IEEE Intelligent Vehicles Symposium, pp. 894–899 (2007)

[3] Efe M. O.: Robust Low Altitude Behavior Control of a Quadrotor Rotorcraft through Sliding Modes. In: Mediterranean Conf. on Control and Automation (2007)

[4] Bouadi H., Bouchoucha M., Tadjine M.: Sliding Mode Control Based on Backstepping Approach for an UAV Type Quadrotor. In: World Academy of Science, Engineering and Technology (2007)

[5] Madani T., Benallegue A.: Backstepping Control for a Quadrotor Helicopter. In: International Conference on Intelligent Robots and Systems, pp. 3255–3260 (2006)

[6] Maciejowski J. M.: Predictive Control with Constraints. Prentice Hall, October (2000)

[7] Lee J. H.: Model Predictive Control: Review of the Three Decades of Development. In: International Journal of Control, Automation, and Systems, Volume 9, Number 3, pp. 415–424 (2011)

[8] Kale M. M., Chipperfield A. J.: Reconfigurable Flight Control Strategies Using Model Predictive Control. In: IEEE International Symposium on Intelligent Control, Taiwan, pp. 43–48 (2002)

[9] Gibbens P. W., Medagoda E. D. B.: Efficient Model Predictive Control for Aircraft. In: Journal of Guidance, Control, and Dynamics, Volume 34, Number 7, pp. 1909–1915 (2011)

[10] Liu Y. C.: Model Predictive Control with Integral Control and Constraint Handling for Mechatronic Systems. In: International Conference on Modelling, Identification and Control, Taiwan, pp. 424–429 (2010)

[11] Murray R. M.: Optimization-based Control. In: California Institute of Technology (2010)

[12] Wang L.: Model Predictive Control System Design and Implementation Using MATLAB. Springer-Verlag London Limited (2009)

[13] Rawlings J. B.: Tutorial: Model Predictive Control Technology. In: Proceedings of the American Control Conference, June (1999)

[14] Lai L. C., Yang C. C., Wu C. J.: Time-Optimal Control of a Hovering Quad-Rotor Helicopter. In: Journal of Intelligent and Robotic Systems, pp. 115–135 (2006)

[15] Quanser's Rapid Control Prototyping (QuaRC). Last accessed 01 Aug 2012. [Online]. Available: *//www.quanser.com/quarc*

[16] NaturalPoint Product Information Manual. Last accessed 01 Aug 2012. [Online]. Available: *http://www.naturalpoint.com/*

[17] Fletcher R.: Practical Methods of Optimization. Second Edition, John Wiley & Sons, (1987)

[18] Dimitrov D., Wieber P. B., Stasse O., Ferreau H. J., Diedam H.: An Optimized Linear Model Predictive Control Solver. In: Springer, pp. 309–318, (2010)

[19] Soldiers Magazine (2007). Last accessed 01 Aug 2012. [Online]. Available: *http://findarticles.com/p/articles/*

[20] Zometa P., Kogel M., Faulwasser T., Findeisen R.,: Implementation Aspects of Model Predictive Control for Embedded Systems. In: American Control Conference (2012)

[21] Isermann R.: Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance. Springer, (2006)

[22] Maciejowski J. M., Jones C. N.: MPC fault-tolerant flight control case study: flight 1862. In: IFAC Symp. on Safeprocess, June (2003)

[23] Maciejowski J. M.: Fault-tolerant aspects of MPC. In: IEE Seminar on Practical Experiences with Predictive Control, February, (2000)

[24] Blanke M., Kinnaert M., Lunze J., Staroswiecki M.: Fault-Diagnosis Systems: Diagnosis and fault-tolerant control. Springer, (2006)

[25] Ogata K.: Modern Control Engineering. Prentice-Hall,Inc., 4th Edition, (2002)

[26] Nelson R. C.: Flight Stability and Automatic Control. McGraw-Hill,Inc., 2nd Edition, (1998)

[27] Borrelli F., Bemporad A., Morari M.: Predictive Control for Linear and Hybrid Systems, (2012)

[28] Bresciani T.: Modelling, Identification and Control of a Quadrotor Helicopter. Master's Thesis, Department of Automatic Control, Lund University, (2008)