Model Predictive Control with Fault Detection and Diagnosis for Multivariable Systems

Vinayak Deshpande

A Thesis

in

The Department

of

Mechanical, Industrial and Aerospace Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Mechanical Engineering) at

Concordia University

Montréal, Québec, Canada

May 2021

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: VINAYAK DESHPANDE

Entitled: Model Predictive Control with Fault Detection and Diagnosis for Multivariable Systems

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

MASTER OF APPLIED SCIENCE - MECHANICAL ENGINEERING

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

Signed by the final examining committee:

_____ Chair
DR. WEN-FANG XIE

_____ Examiner
DR. JUN YAN

_____ Examiner
DR. WEN-FANG XIE

_____ Thesis Supervisor(s)
DR. YOUMIN ZHANG

_____ Thesis Supervisor(s)

Approved by _____
DR. SIVAKUMAR NARAYANSWAMY Chair of Department or Graduate Program Director

_____
Dean MOURAD DEBBABI

Abstract

Model Predictive Control with Fault Detection and Diagnosis for Multivariable Systems

Vinayak Deshpande

The feedback control system design technique of Model Predictive Control (MPC) has been vastly used in the chemical and process engineering industry, due to its ability to handle dynamics with multiple inputs and multiple outputs, which are essentially the majority of today's engineering systems. In addition, the field of Fault Detection and Diagnosis (FDD) in control systems also has been extensively researched over the past decades as it is critical for the controller to realize when and if a fault has occurred within a system. However, due to the high computational requirements, it is often challenging to implement FDD based MPC algorithms in resource limited real world systems. This thesis addresses the development of MPC algorithms with combined state and fault estimation. Firstly, a novel Quadratic Programming (QP) formulation is developed for a recently proposed efficient MPC method along with simultaneous state and fault estimation. Another contribution is the enhancement of a standard integral action MPC algorithm (which has an implicit fault tolerance capability), to provide state and actuator fault estimation in real time. This work focuses on faults which are modeled as a Loss Of Effectiveness (LOE). The algorithm to estimate the system faults and states simultaneously is a simple observer based method which can be tuned beforehand, thus eliminating the need for on-line real time complex calculations. Lastly, a third contribution of this thesis is the application of the above methods to design MPC based flight control systems for fixed wing aircraft. Simulations are presented to demonstrate the effectiveness of the proposed methods.

Acknowledgments

There are several people who I owe the successful completion of this thesis to. I would like to begin by expressing my sincere thanks to my supervisor, Prof. Youmin Zhang for the opportunity to study at Concordia University and pursue my Master's degree in the topic of my choosing under his supervision. His knowledge in the field is unparalleled and his motivation for me to study a brand new topic; Model Predictive Control, is something which I will be forever grateful for as I progress in my engineering career. Also, thank you to Ms. Leslie Hosein, Ms. Charlene Wald and Ms. Marwa Gouda, for the advising and constant administrative support!

During my studies at Concordia and first time living solo in the city of Montréal, and also experiencing the absolutely terrible COVID-19 pandemic, there are several individuals who always brought positivity by my side. Thank you David Venuto (from MILA-McGill) for teaching me how to use the oven! the trips to Québec City, Vermont, St. Laurent Blvd, and always being there whenever I faced any personal hardship or difficulty. Thank you Shirin Ahmadi, Mikhail Mascarenhas, Noel Peter D'Souza and Nikolai Nazarov (from Concordia) for making my stay in Montreal outstanding and for the exploration around the city! Thank you Maddie W. and Aiden from my building for the living support. In addition, thank you to the awesome staff at Concordia University Le Gym and Econofitness - Promenades Cathédrale, where it was an enjoyable experience to work out.

Most importantly, a big thank you to my Mom and Dad, Janardan and Mayura Deshpande, for always encouraging me to pursue graduate studies during my final year at McMaster University. I was fortunate that you were only six hours away in Toronto, so I could come and stay with you during the lockdown in Canada. Thank you Pa for visiting me in Montreal every month and thank you Ma for teaching me how to cook delicious chicken and fish recipes! It would have been impossible for

me to complete my thesis without the constant support from you both. With tremendous pride and honor I dedicate this work to my family and those closest to me.

This work is supported by the Concordia Graduate Scholarship in Natural Sciences and Engineering and the Natural Sciences and Engineering Research Council - Canada Graduate Scholarship Master's awards.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Multivariable Constrained Systems

The vast majority of today's engineering systems have operational limits and also possess multiple inputs and outputs. This classifies them as MIMO (multi-input-multi-output) systems. A simple example is a car; the inputs include steering wheel turn angle, gear setting, brake setting, and outputs include vehicle velocity, orientation, and tire friction. An illustration of a MIMO system is provided here.

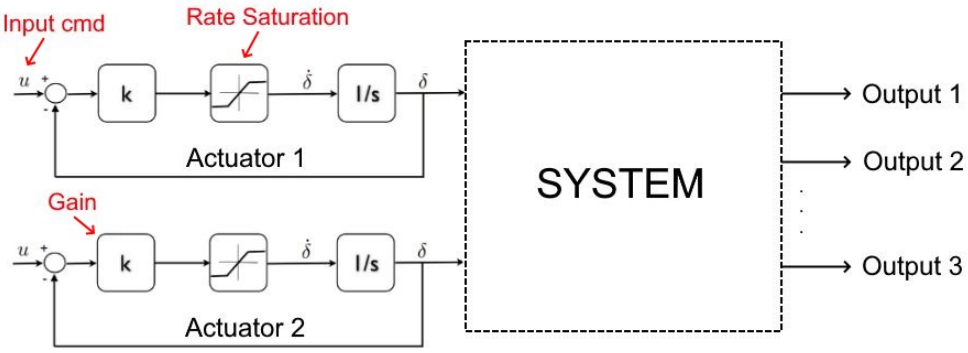

Figure 1.1: MIMO System with Constrained Actuators

The application of the developed MPC models in this thesis will focus on Unmanned Aerial Vehicles (UAVs), particularly fixed wing aircraft. In general, UAVs have very complex dynamic behaviour. There is a strong amount of coupling present, between the various input control surfaces and the

response of these vehicles, for each output. As a result, control system design for UAVs begins with linearizing the nonlinear dynamic equations of motion and then decoupling them into longitudinal and lateral modes [1, 2]. When considering the longitudinal and lateral dynamics separately, these are still multivariable in nature as more than one control surface is involved in each of these modes. In addition, the actuators on-board an aircraft such as spoilers, elevator, ailerons etc. have operational limits as highlighted in [3]. The limits are in the form of 1) Position constraints, where the actuator maximum and minimum position is bounded and 2) Rate constraints, where the incremental actuator movement (i.e motion at each moment in time) is bounded. Both constraints are equally critical since if any are exceeded, it can result in mechanical damage to the actuator and/or the UAV. A schematic of a MIMO constrained system is illustrated in Figure 1.1.

For a MIMO system, each output will typically have a different response time with respect to each input. As a result, it would be useful to utilize an MPC method which attempts to decouple these outputs in relation to each input, so that system performance can be maximized.

1.2 Model Predictive Control

There are multiple methods to design controllers and a large amount of research has been performed in linear and nonlinear flight control system design. For instance, the well known Proportional Integral Derivative (PID) control has been used extensively as per [4, 5]. However, this control method does not work very well for MIMO systems as the individual gains have to be tuned perfectly, and it also does not account for the actuator rate constraints. Using more advanced nonlinear methods such as sliding mode or backstepping control [6, 7] are very complex to implement, and also do not take actuator constrains into account.

Model Predictive Control (MPC) is a method where at each time step, the optimal system inputs are calculated and assigned based on constraints in real time, via a standard Quadratic Programming (QP) procedure. These inputs are based on the predicted system outputs over a specified time in the future, known as the 'prediction horizon'. The concept of Receding Horizon Control (RHC) is to only apply the first obtained set of inputs. The prime benefit of MPC is it's ability to account for all kinds of constraints, and the fact that it is ideal for MIMO systems as explained in [8]. The standard

MPC method is the 'Integral-Action' method, where the incremental input is calculated [9]. However, recent advancements in MPC have led to faster efficient algorithms where the absolute input is calculated and assigned [10]. Typically, internal MPC models consist of very large matrices and this requires a large amount of processing time. Furthermore, the quadratic programming solvers calculate the optimal input through an iterative process at each time step. Due to these reasons, has not been used widely in aerospace systems, especially for commercial applications.

It will be later seen that the efficient MPC method provides a few advantages in that it 1) the matrices are of much smaller dimension as they are based on the state transition matrix and 2) it solves for the optimal absolute input rather than the incremental input. This study aims to develop a constrained formulation for the efficient MPC method and adjust the control law in the presence of actuator faults. However, the Integral-Action MPC method provides a degree of implicit fault tolerance capability as highlighted in [11], unlike the efficient MPC method. In this study, two QP solvers will be utilized. Neither involves calculating matrix inverses, which is a cumbersome process. The first solver is the well known Hildreth's Quadratic Programming (HQP) procedure which uses active set methods [12]. The second algorithm, a Parallel Quadratic Programming (PQP) algorithm, first proposed by Brand et al. [13], uses iterative multiplication. If the PQP algorithm is run on a parallel platform, the solver time can be reduced significantly. Moreover, the PQP algorithm will be faster to implement on hardware since it is only a single equation, whereas the HQP algorithm requires an additional maximum element search at each iteration. These above factors need to be considered when choosing solver to be implemented on a flight computer.

1.3 Fault Detection and Diagnosis

The technology of today is largely automated in nature and does not require a significant amount of intervention during operation. Examples include self driving cars and aircraft autopilots. This inherently poses a risk for unexpected software and hardware failures. In the majority of situations, vehicle operators such as pilots and drivers do not have sufficient knowledge or the time to diagnose sudden failures. Hence, there is a growing need for algorithms to automatically detect and diagnose faults before occurrence of a catastrophe. This increased reliance on autonomy has led to ongoing

research in the field of fault tolerant control design [14]. In order to prevent a catastrophe, the ability of any control algorithm to detect a fault is critical. Various types of faults could occur such as sensor, actuator, and faults in the internal plant model. This thesis will focus on actuator faults, which typically involve a Loss Of Effectiveness (LOE) [15]. For instance, if a spoiler is commanded to deploy to 75% of it's maximum, a 50% LOE indicates that the spoiler will behave as if it is only deployed to 37.5% of it's maximum. The least complex fault detection algorithms are typically observer based [16].

1.4 Motivation & Thesis Organization

Since MPC is a promising solution to multivariable constrained systems and considering the importance of Fault Detection and Diagnosis, this study aims to combine these two methods to develop novel low complexity MPC algorithms for linear-time-invariant (LTI) systems. Chapter 2 provides the detailed theory of the two MPC algorithms in question. Chapter 3 describes the method used to detect actuator faults. Chapter 4 provides the aircraft mathematical models used for the MPC application. Chapter 5 illustrates the performance of the developed MPC methods and finally Chapter 6 draws conclusions obtained and outlines possible topics for future study.

1.4.1 Accepted Publications

- **Vinayak Deshpande** and Youmin Zhang, "Multivariable Receding Horizon Control of Aircraft with Actuator Constraints," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2020, (pp. 1846-1851).

- **Vinayak Deshpande** and Youmin Zhang, "Integral Action Model Predictive Control with Actuator Fault Estimation," Accepted by 4^{th} *IEEE International Conference on Industrial Cyber-Physical Systems (ICPS'21)*. (May 10-13, 2021)

1.4.2 Submitted Publications

- **Vinayak Deshpande** and Youmin Zhang, "Fault Tolerant Model Predictive Control of Unmanned Aircraft with Actuator Fault Estimation," To be submitted to *Guidance Navigation and Control (GNC): World Scientific Publishing Co*. (2021)

- **Vinayak Deshpande** and Youmin Zhang, "Model Predictive Control of Fixed Wing Aircraft using a Disturbance Observer Approach," Submitted to *ASME/IEEE International Conference on Mechatronic and Embedded Systems and Applications*, SUAVTA 2021: Symposium on Unmanned Autonomous Vehicle Technologies and Applications. (August 17-20, 2021)

Chapter 2

Model Predictive Control

2.1 Mathematical Preliminaries

The following notation will be used throughout this thesis: R_+ denotes positive real numbers, $\mathbb{R}^n$ denotes $n \times 1$ real column vectors, $\mathbb{R}^{n \times m}$ denotes $n \times m$ real matrices, $\hat{X}$ is an estimate of the vector $X \in \mathbb{R}^n$, $\mathcal{L}$ represents the Lagrange operator, $A_{i,j}$ denotes the ijth element of the matrix A, $A[i]$ is the i^{th} row of matrix A, R_i represents the ith element of the column R, $A \otimes B$ is the Kronecker product of A, B. I_n represents the $n \times n$ identity matrix, A^T, A^{-1} represents the transpose and inverse of matrix A. $Cov(v)$ is the covariance of v. A Gaussian normal distribution with mean μ and variance σ^2 is represented by $\mathcal{N}(\mu, \sigma^2)$, and sgn, dim represent the sign and dimension functions, respectively. An $n \times m$ zero matrix, $n \times 1$ zero column vector, $1 \times n$ zero row vector is given by $0_{n \cdot m}, 0^n, 0_n$ respectively.

2.2 MIMO State Space Models

Using a discrete-time setting where k being the time step, Linear Time Invariant (LTI) systems are considered. The nominal model with faulty actuators is given by:

$$X_{k+1} = AX_k + B_f U_k + w_k \tag{1}$$

$$Y_k = CX_k + v_k \tag{2}$$

$$Z_k = HY_k \tag{3}$$

For this discrete system, the equivalent continuous time model is given by:

$$X_{k+1} = A_c X_k + B_{cf} U_k + w_k \tag{4}$$

$$Y_k = C X_k + v_k \tag{5}$$

$$Z_k = HY_k \tag{6}$$

where the states are given by $X \in \mathbb{R}^n$, inputs are $U \in \mathbb{R}^m$, outputs are $Y \in \mathbb{R}^p$. If $p > 1$ and $m > 1$, the system is considered MIMO. The discretization time is dT and should be as small as possible to preserve accuracy. The controlled variables (CVs) are $Z \in \mathbb{R}^r$. That is to say, the setpoints are defined for Z. The process and measurement noise is represented by $w \in \mathbb{R}^n$ and $v \in \mathbb{R}^p$ respectively, having covariances $Q \in \mathbb{R}^{n \times n}$ and $R \in \mathbb{R}^{p \times p}$. The faulty input matrix $B_f \in \mathbb{R}^{n \times m}$ is given by:

$$B_f = B(I_m + diag(f)) = B_c(I_m + diag(f)) \tag{7}$$

where $B, (B_c)$ is the nominal system discrete, (continuous) input matrix, and the fault parameter vector $f \in \mathbb{R}^m \rightarrow [f_1 \dots f_m]^T$ represents the loss of effectiveness having a valid range of: $-1 \leq f_i \leq 0$. Hence, when $f_i = -1$ it represents a 100% LOE and when $f_i = 0$, the actuator is fault-free.

2.3 An Efficient MPC Method

The efficient model predictive control algorithm was first proposed in [10]. This algorithm directly solves for the absolute system input U (whilst considering all constraints), at each time step k, and it uses the state transition matrix ($\phi \in \mathbb{R}^{n \times n}$) to solve for the predicted outputs or CVs. A visual representation of this method is provided in Fig. 2.1.

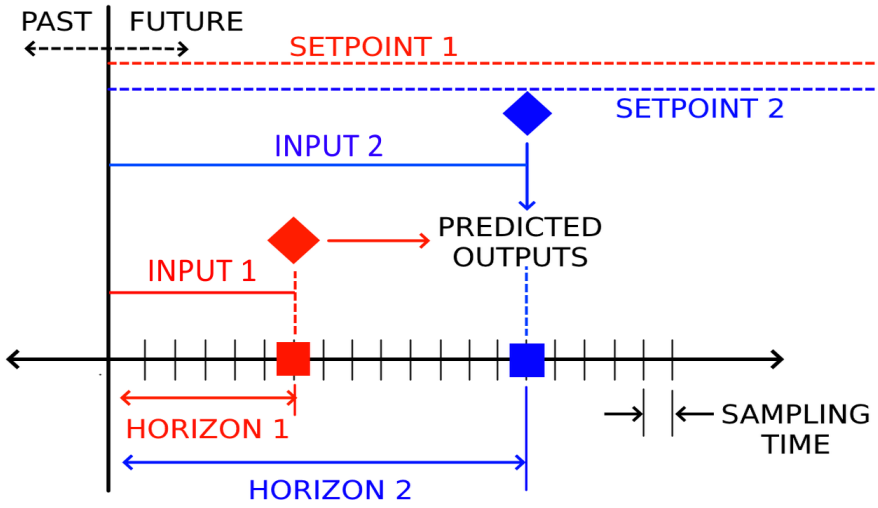

Figure 2.1: Single Point Efficient MPC Algorithm - 2 Horizons and CVs

A unique advantage of the efficient algorithm is that different prediction horizons can be defined for each CV / output as emphasized in [17]. The matrix ϕ is formed from the equivalent continuous time system matrix A_c and is given by:

$$\phi(\delta t_{i+1} - \delta t_i) \triangleq e^{A_c(\delta t_{i+1} - \delta t_i)} \triangleq S \cdot \begin{bmatrix} e^{\lambda_1(\delta t_{i+1} - \delta t_i)} & 0 & \dots & 0 \\ 0 & e^{\lambda_2(\delta t_{i+1} - \delta t_i)} & & \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & e^{\lambda_n(\delta t_{i+1} - \delta t_i)} \end{bmatrix} S^{-1} \quad (8)$$

where the length of the prediction horizon is given by $(\delta t_{i+1} - \delta t_i) \in \mathbb{R}_+$. Clearly, the system must not be ill-conditioned otherwise in that case, no MPC algorithm would be closed loop stable. For this study, to reduce complexity, each CV (Z_k) is chosen to be predicted forward rather than each output (Y_k), as all outputs may or may not be controlled (i.e. $r \leq p$).

To proceed with MPC, a unique prediction horizon for each CV $N_{p_i} \in \mathbb{R}_+$ is specified by: $\forall i \in [0, r]$. If a single prediction point is used for each output, the control horizon N_c is unity. By partitioning H into $[H_i \dots H_r]^T$, the CVs Z_k are split into $[Z_i \dots Z_r]^T$.

The first step of MPC is to solve for the predicted control variables $Z_{pr} \in \mathbb{R}^r$ at each time step k. For r control variables, the index $i \in [1, r]$. If multiple prediction points n_p are used, the index $j \in [1, n_p]$. For the efficient MPC method, the predicted CVs can then be expressed from the current state and future control inputs as follows:

$$Z_{pri,j} = H_i C\phi(\delta_{ti,j})\hat{X} + \sum_{k=1}^{j} H_i C\phi(\delta_{ti,j} - \delta_{ti,k})\Gamma(\delta_{ti,k} - \delta_{ti,k-1}) \tag{9}$$

If a single prediction point is used for each CV, which was the case throughout this study, then $j = 1$, and (9) can be simplified to:

$$Z_{pri} = H_i C\phi\underbrace{(\delta_{ti})}_{N_{pi}}\hat{X} + H_i C\Gamma\underbrace{(\delta_{ti})}_{N_{pi}} \tag{10}$$

Then, (10) can be broken up and further simplified into the following terms

$$Z_{pr} = F\hat{X}_k + GU_k \tag{11}$$

$$\text{where } F \triangleq \begin{bmatrix} H_1 C\phi(N_{p_1}) \\ H_2 C\phi(N_{p_2}) \\ \vdots \\ H_r C\phi(N_{p_r}) \end{bmatrix} \tag{12}$$

$$\text{and } G \triangleq \begin{bmatrix} H_1 C\Gamma(N_{p_1}) \\ H_2 C\Gamma(N_{p_2}) \\ \vdots \\ H_r C\Gamma(N_{p_r}) \end{bmatrix} \tag{13}$$

The matrices $F \in \mathbb{R}^{r\times n}$ and $G \in \mathbb{R}^{r\times m}$ can either be constant or defined at each interval k, for time varying prediction horizons i.e. a changing N_{pi} at each time step. The matrix $\Gamma \in \mathbb{R}^{n\times m}$ is solved from the equivalent continuous time model and is given by:

$$\Gamma(\delta t_{i+1} - \delta t_i) = A_c^{-1}[\phi(\delta t_{i+1} - \delta t_i) - I_n]B_c \tag{14}$$

In MIMO systems, it is well known that when $m \leq r$, it is only possible to track maximum m CVs. However, for all simulations in this study either it was the case that $m = r$ or $m > r$. Defining the setpoint vector as $\mathcal{R} \in \mathbb{R}^r$, which contains the desired value for each CV, the error vector $E \in \mathbb{R}^r$ can then be calculated:

$$E = \mathcal{R} - F\hat{X}_k \tag{15}$$

The optimal control input U is solved by evaluation of a cost function. Using the standard Linear Quadratic Regulator (LQR) method and substituting (11), the objective function can be defined as in [18]:

$$(\underset{\mathbf{U}}{min})J = U^T HU + 2f^T U + E^T QE \tag{16}$$

Here $H \triangleq G^T QG + R$ and $H > 0, H \in \mathbb{R}^{m\times m}$. The term $f \triangleq -G^T QE$ and $f \in \mathbb{R}^m$. The matrices $Q \in \mathbb{R}^{r\times r}$, $Q > 0$ and $R \in \mathbb{R}^{m\times m}$, $R > 0$ are positive definite symmetric matrices representing weights on the CV error and control action respectively. The symmetric Q, R matrices will then result in $H = H^T$. Setting $\partial J/\partial U = 0$, the unconstrained solution is obtained as:

$$U = H^{-1} f = KE \tag{17}$$

The feedback gain $K \in \mathbb{R}^{m\times r}$ is used to solve for the unconstrained optimal input U. Based on K the closed loop system matrix A_{cl} can be determined. The corresponding equations are given by:

$$K = (G^T QG + R)^{-1} G^T Q \tag{18}$$

$$A_c = A - BKF \tag{19}$$

where $Q \in \mathbb{R}^{r\times r}$ and $R \in \mathbb{R}^{m\times m}$ are the weight matrices on the CVs and the inputs respectively. This is analogus to the weighting matrices used in the LQR control design method. If the unconstrained U from (17) exceeds the input constraints, the quadratic optimization procedure is activated to resolve for the constrained optimal inputs U, and that is assigned to the system. A novel QP procedure was developed by the author in [19], and a full derivation, along with the QP solvers used in this study is provided in Appendices A and B.

2.4 An Integral Action MPC Method

This is the traditional well known MPC method which solves for the optimal incremental input ΔU. A visual representation is provided in Figure 2.2 [20].

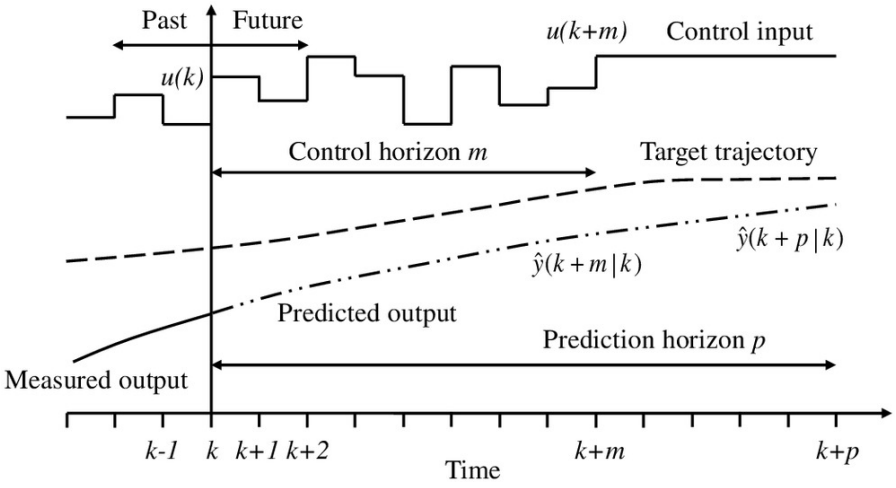

Figure 2.2: Integral Action MPC Algorithm

This time, only a single prediction and control horizon $\hat{N}_p, \hat{N}_c \in \mathbb{R}_+$ is assigned where $\hat{N}_c \leq \hat{N}_p$. However, the definition of the predicted CVs Z_{pr} is more complex because multiple points are used, unlike the efficient MPC method where only one prediction point is present (which covers a longer time interval).

To explain further, let's compare a single point efficient MPC (*EMPC*) and integral action MPC (*IMPC*). For simplicity, let's also state that for *EMPC*, the prediction horizon is the same for all CVs (i.e $N_{p1} = N_{p2} = \ldots N_{pr}$). Table 2.1 provides a comparison.

Table 2.1: Prediction Points - Efficient vs Integral Action MPC

Parameter	**Description**	**EMPC**	**IMPC**
dT	Simulation Sampling Time	0.01	0.05
N_p (all CVs)	Prediction Horizon (s)	3.00	1.50
N_c	Control Horizon (s)	$dT = 0.01$	$2 \cdot dT = 0.10$
True N_p (all CVs)	Number of Prediction Points (for Z_k)	1	30
True N_c	Number of Control Points (for U_k)	1	2

Hence, it is evident that the number of prediction points is larger for the integral action method (based on the time step size). They are evenly spaced from the current time k to the end of the

horizon $(k+1, k+2 \ldots k+N_p/dT)$ - as also seen in Fig. 2.2. For this MPC method, the predicted CVs Z_{pr} can then be expressed from the current state and future control inputs as follows. A complete derivation is provided within [21].

$$\begin{aligned}
Z_{pr(k+1)} &= \bar{H}\bar{A}\bar{X}_k + \bar{H}\bar{B}\Delta U_k \\
Z_{pr(k+2)} &= \bar{H}\bar{A}^2\bar{X}_k + \bar{H}\bar{A}\bar{B}\Delta U_k + \bar{H}\bar{B}\Delta U_{k+1} \\
Z_{pr(k+3)} &= \bar{H}\bar{A}^3\bar{X}_k + \bar{H}\bar{A}^2\bar{B}\Delta U_k + \bar{H}\bar{A}\bar{B}\Delta U_{k+1} + \bar{H}\bar{B}\Delta U_{k+2} \\
&\vdots \\
Z_{pr(k+N_p)} &= \bar{H}\bar{A}^{N_p}\bar{X}_k + \bar{H}\bar{A}^{N_p-1}\bar{B}\Delta U_k + \ldots + \bar{H}\bar{A}^{N_p-N_c}\bar{B}\Delta U_{k+N_c-1}
\end{aligned} \tag{20}$$

Similar to the efficient MPC method, (20) can be simplified to:

$$Z_{pr} = F_z\hat{\bar{X}}_k + G_z\Delta U_k \tag{21}$$

where $\hat{X}$ represents the state estimate and the matrices $F_z \in \mathbb{R}^{(N_p \cdot r)\times(n+r)}$ and $G_z \in \mathbb{R}^{(N_p \cdot r)\times(m \cdot N_c)}$ are used to determine the predicted CVs and should be calculated ahead of time as they are usually. The ΔU_k term in (21) is expressed across the entire control horizon i.e. $\Delta U = [\Delta U_k,\ \Delta U_{k+1},$ $\Delta U_{k+2} \ldots \Delta U_{k+N_c-1}]^T$, because it may not be unity unlike the efficient MPC. Next, the matrix F_z is given by:

$$F_z \triangleq \begin{bmatrix} \bar{H}\bar{A} \\ \bar{H}\bar{A}^2 \\ \bar{H}\bar{A}^3 \\ \vdots \\ \bar{H}\bar{A}^{N_p} \end{bmatrix} \tag{22}$$

and the matrix G_z is given by:

$$G_z \triangleq \begin{bmatrix} \bar{H}\bar{B} & 0_{r\times m} & 0_{r\times m} & 0_{r\times m} \\ \bar{H}\bar{A}\bar{B} & \bar{H}\bar{B} & 0_{r\times m} & 0_{r\times m} \\ \bar{H}\bar{A}^2\bar{B} & \bar{H}\bar{A}\bar{B} & \bar{H}\bar{B} & 0_{r\times m} \\ \vdots & \vdots & \vdots & \vdots \\ \bar{H}\bar{A}^{N_p-1}\bar{B} & \bar{H}\bar{A}^{N_p-2}\bar{B} & \bar{H}\bar{A}^{N_p-3}\bar{B} & \bar{H}\bar{A}^{N_p-N_c}\bar{B} \end{bmatrix} \tag{23}$$

The terms $\bar{H} \in \mathbb{R}^{r\times(n+r)}$, $\bar{A} \in \mathbb{R}^{(n+r)\times(n+r)}$, and $\bar{B} \in \mathbb{R}^{(n+r)\times m}$ are formed from the nominal discrete model and are given by:

$$\bar{H} \triangleq [HC \;\; I_r] \tag{24}$$

$$\bar{B} \triangleq \begin{bmatrix} B \\ 0_{r\times m} \end{bmatrix} \tag{25}$$

$$\bar{A} \triangleq \begin{bmatrix} A & 0_{n\times r} \\ HC & I_r \end{bmatrix} \tag{26}$$

The augmented estimated state vector $\hat{\bar{X}} \in \mathbb{R}^{n+r}$ containing both the incremental states along with the CV output values, is given by:

$$\hat{\bar{X}} \triangleq \begin{bmatrix} \hat{X}_k - \hat{X}_{k-1} \\ HC\hat{X}_k \end{bmatrix} \tag{27}$$

Once again, Defining the setpoint vector as $\mathcal{R} \in \mathbb{R}^{N_p\times r}$, the error vector $\bar{E} \in \mathbb{R}^{N_p\times r}$, which is clearly spread out over the entire prediction horizon, can then be formulated as:

$$\bar{E} = \mathcal{R} - F_z\hat{X}_k \tag{28}$$

In a similar fashion to the efficient MPC algorithm, this time the cost function for the Integral Action method has the incremental optimal input ΔU as the decision variable and is expressed as:

$$\underset{\Delta U}{(min)} J = \Delta U^T \mathcal{H} \Delta U + 2\bar{f}^T U + E^T \bar{Q} E \tag{29}$$

In (29), the term $\mathcal{H} \triangleq G_z^T \bar{Q} G_z + \bar{R}$, and $\mathcal{H} \in \mathbb{R}^{(m \cdot N_c) \times (m \cdot N_c)}$. The term $\bar{f} \triangleq -Gz^T \bar{Q} \bar{E}$ and $\bar{f} \in \mathbb{R}^{m \times N_c}$. Two positive definite matrices $Q \in \mathbb{R}^{r \times r}$ and $R \in \mathbb{R}^{m \times m}$ are defined as weighting matrices on the CVs and the control action, respectively. Next, they are augmented into $\bar{Q}, \bar{R}$ by the expressions: $\bar{Q} = I_{N_p} \otimes Q$ and $\bar{R} = I_{N_c} \otimes R$, to express the weight factors over the entire prediction and control horizons. The setpoint vector $\mathcal{R} \in \mathbb{R}^{N_p \times r}$ is also defined over the prediction horizon.

The unconstrained optimal solution for ΔU^* at each time step, based on the Receding Horizon principle, is to select the first calculated m elements of:

$$\Delta U^* = \mathcal{H}^{-1} G_z^T \bar{Q}(\bar{E}) = \bar{K} E \tag{30}$$

Similar to the efficient MPC method, the feedback gain $\bar{K} \in \mathbb{R}^{(m \cdot N_c) \times (N_p \cdot r)}$ is used to solve for the unconstrained optimal input ΔU. Once again, based on $\bar{K}$ the closed loop system matrix A_{cl} can be determined, this time the first $(m \times n)$ elements can be extracted from $\bar{K} F_z$. The corresponding equations are given by:

$$\bar{K} = (G_z^T \bar{Q} G_z + \bar{R})^{-1} G_z^T \bar{Q} \tag{31}$$

$$K = [\bar{K} F_z]_{1:m,\ 1:n} \tag{32}$$

$$A_{cl} = A - BK \tag{33}$$

If the unconstrained solution ΔU from (30) exceeds the incremental/ absolute input constraints, the quadratic optimization procedure is activated to resolve for the constrained optimal inputs ΔU, and that is assigned to the system. The QP procedure and solvers are identical to that used in the efficient MPC method, since the definition of each cost function is practically equivalent. The Integral Action MPC cost function is just expressed in terms of ΔU instead of U. In addition, it's dimension is larger, as opposed to using single point predictions in the efficient MPC method.

2.5 Realization of Constraints

Constraints on actuator position U and incremental movement ΔU are expressed as:

$$U_{min} \leq U \leq U_{max} \tag{34}$$

$$\Delta U_{min} \leq \Delta U \leq \Delta U_{max} \tag{35}$$

Incremental actuator motion is calculated by $\Delta U_k \triangleq U_k - U_{k-1}$. Next, the constraints for the efficient MPC algorithm can be compactly expressed together as:

$$MU \leq \gamma \tag{36}$$

This is because the objective variable is U. However, for the integral action MPC algorithm, the constraints are expressed as:

$$\bar{M}\Delta U \leq \bar{\gamma} \tag{37}$$

For the efficient MPC method, assuming a unity control horizon N_c, The term $M \in \mathbb{R}^{4m \times m}$ contains indices $(1, -1, 0)$ for position and rate constraints (or $M \in \mathbb{R}^{2m \times m}$ if either position or rate constraints are accounted for), and $\gamma \in \mathbb{R}^{4m}$ contains the numerical values of these constraints (or $\gamma \in \mathbb{R}^{2m}$ if either position or rate constraints are accounted for). Note that it is easily possible to consider both absolute and incremental constraints whether (36), or (37) is used. To illustrate, let us assume that there are 2 actuators with active position and rate constraints i.e. $(m = 2)$, maximum constraints are > 0 and minimum constraints are < 0, (36) would be expressed as:

$$\underbrace{\begin{bmatrix} sgn(U_{1_{max}}) & 0 \\ sgn(U_{1_{min}}) & 0 \\ 0 & sgn[(U_{2_{max}})] \\ 0 & sgn[(U_{2_{min}})] \\ sgn[\Delta(U_{1_{max}})] & 0 \\ sgn[\Delta(U_{1_{min}})] & 0 \\ 0 & sgn[\Delta(U_{2_{max}})] \\ 0 & sgn[\Delta(U_{2_{min}})] \end{bmatrix}}_{M} \begin{bmatrix} U_{1_k} \\ U_{2_k} \end{bmatrix} \leq \underbrace{\begin{bmatrix} U_{1_{max}} \\ -U_{1_{min}} \\ U_{2_{max}} \\ -U_{2_{min}} \\ \Delta(U_{1_{max}}) + U_{1_{k-1}} \\ -\Delta(U_{1_{min}}) - U_{1_{k-1}} \\ \Delta(U_{2_{max}}) + U_{2_{k-1}} \\ -\Delta(U_{2_{min}}) - U_{2_{k-1}} \end{bmatrix}}_{\gamma} \tag{38}$$

For the integral action MPC, the control horizon N_c may not necessarily be unity. In this case, the definition of $\bar{M}$, ΔU and $\bar{\gamma}$ becomes more complex. Firstly, $\forall k \rightarrow \bar{M} \in \mathbb{R}^{(4m \cdot N_c) \times (m \cdot N_c)}$, $\bar{\gamma} \in \mathbb{R}^{4m \cdot N_c}$ and $\Delta U \in \mathbb{R}^{m \cdot N_c}$. If either position or rate constraints are accounted for, then $4m \rightarrow 2m$. In other words, there are N_c copies created to map the constraints across the entire control horizon. The identity matrices are present because once again the indices are ± 1 or 0 based on the sign of the max/min constraint. Generally, identity matrices either on their own or in lower triangular form are used within the $\bar{M}$ term and the complete methodology is demonstrated in [21]. To illustrate, let us assume that there are 2 actuators with active position and rate constraints, and their max constraints are > 0 and min constraints are < 0. In this case, (37) would be expressed as:

$$\underbrace{\begin{bmatrix} \begin{bmatrix} I_{m\cdot N_c} \end{bmatrix} \\ \\ \begin{bmatrix} -I_{m\cdot N_c} \end{bmatrix} \\ \\ \underbrace{\begin{bmatrix} I_m & 0 \\ I_m & I_m \\ \vdots & \end{bmatrix}}_{N_c\ rows} \\ \\ \underbrace{\begin{bmatrix} -I_m & 0 \\ -I_m & -I_m \\ \vdots & \end{bmatrix}}_{N_c\ rows} \end{bmatrix}}_{\bar{M}} \underbrace{\begin{bmatrix} \begin{bmatrix} U_{1_k} \\ U_{2_k} \end{bmatrix} \\ \vdots \end{bmatrix}}_{N_c\ copies} \leq \underbrace{\begin{bmatrix} \underbrace{\begin{bmatrix} \Delta(U_{1max}) \\ \Delta(U_{2max}) \\ -\Delta(U_{1min}) \\ -\Delta(U_{2min}) \end{bmatrix}}_{N_c\ copies} \\ \vdots \\ \underbrace{\begin{bmatrix} U_{1max} - U_{1k} \\ U_{2max} - U_{2k} \end{bmatrix}}_{N_c\ copies} \\ \vdots \\ \underbrace{\begin{bmatrix} -U_{1min} + U_{1k} \\ -U_{2min} + U_{2k} \end{bmatrix}}_{N_c\ copies} \end{bmatrix}}_{\bar{\gamma}} \tag{39}$$

In conclusion, it can be seen that the matrix $M, \bar{M}$ can be defined beforehand, whereas the term $\gamma, \bar{\gamma}$ has to be updated in real time with each input U_k comes in. Note that if any of the maximum or minimum constraints (for $U, \Delta U$) equal 0, then within $(M, \bar{M})$ the mapping element would have to be changed from $(-1$ or $1)$ to 0 respectively.

2.6 Predictive Control with Disturbance Observers

2.6.1 Disturbance Observer Design

During real life operation, a non zero external disturbance can occur on a system. For instance a car could be driving and there could be a wind turbulence force on the vehicle frame of some magnitude and direction. Another example is an antenna positioning system under the influence of an external force. This disturbance leads to a mismatch (offset) between the intended plant operation and it's true desired operation. As the disturbance is not captured within the system's intended dynamics, the plant must be augmented with the disturbance model. This leads to the problem of offset-free Model

Predictive Control, as mentioned in [22]. It is crucial for any control algorithm (not just MPC based) to correctly estimate this disturbance, so that the reference signal can be adjusted (i.e. offset) and the true plant outputs can then reach the reference signal. This is a form of "disturbance rejection" control. Assuming that not all system states may be necessarily subject to this disturbance, (1) can be rewritten as:

$$X_{k+1} = AX_k + B_f U_k + B_d w_k \tag{40}$$

$$Y_k = CX_k + C_d w_k + v_k \tag{41}$$

In control theory, an "observer" or estimator, is an algorithm/ equation used to estimate an unknown parameter. Observers are generally used to estimate the system states $\hat{X}$. This is because a system equipped with a sensor will only be able to read the noisy outputs Y. In this section, the disturbance w is assumed to be constant, hence (40) is to a degree, a deterministic signal. Within this disturbance w, there is noise present (known as the process noise). To design the observer, an optimal gain matrix must be computed and used. The objective is to now estimate not only the states $\hat{X}$, but also this constant non zero disturbance $\hat{w}$. This is done in the form of a "disturbance observer". Intuitively, an initial guess value of $\hat{X}, \hat{w}$ must be provided. It is assumed that the number of states X affected by w, i.e. n_d is known.

In (40), the disturbance is given by: $w \in \mathbb{R}^{n_d}$, where $w \in \mathcal{N}(\mu, \sigma^2)$ and the mean $\mu \neq 0$. The gain matrices $B_d \in \mathbb{R}^{n \times n_d}$ and $C_d \in \mathbb{R}^{p \times n_d}$ are assumed to be known, and either $n_d \neq n$ or $n_d = n$. Since w represents the constant load disturbance, it has a constant non-zero mean. The term $v \in \mathbb{R}^p \in \mathcal{N}(0, \sigma^2)$ is the measurement noise (captured by the sensor). This is a normal (Gaussian) distribution with zero mean and some variance σ^2. The methodology used to design the observers is taken from [22]. It must be true that $n_d \leq p$, for the observability test to hold. Then, the estimates for both states and disturbances $(\hat{X}, \hat{w})$ can be calculated from the noisy outputs Y and inputs U as:

$$\hat{X}_{k+1} = A\hat{X}_k + B_f U_k + B_d \hat{w}_k + K^x (Y_k - C\hat{X}_k) \tag{42}$$

$$\hat{w}_{k+1} = I_{n_d} \hat{w}_k + K^w (Y_k - C\hat{X}_k) \tag{43}$$

The term $(y_k - C\hat{X}_k)$ represents the innovation or the estimation error, and must be as close to 0 as possible, to guarantee convergence. The gains $K^x \in \mathbb{R}^{n \times p}$ and $K^w \in \mathbb{R}^{n_d \times p}$ are constant, i.e. this is an optimal steady state observer.

2.6.2 Augmented Model

To obtain K^x, K^w the model is first augmented [22] to obtain:

$$\begin{bmatrix} \hat{X} \\ \hat{w} \end{bmatrix}_{k+1} = \underbrace{\begin{bmatrix} A & B_d \\ 0_{n_d \times m} & I_{n_d} \end{bmatrix}}_{\tilde{A}} \begin{bmatrix} \hat{X} \\ \hat{w} \end{bmatrix}_k + \underbrace{\begin{bmatrix} B \\ 0_{n_d \times m} \end{bmatrix}}_{\tilde{B}} + \begin{bmatrix} K^x \\ K^w \end{bmatrix} (Y_k - C\hat{X}_k) \tag{44}$$

$$Y_k = \underbrace{[C \; C_d]}_{\tilde{C}} \begin{bmatrix} \hat{X} \\ \hat{w} \end{bmatrix}_k + v_k \tag{45}$$

In addition, define the process noise covariance terms $Q_n \in \mathbb{R}^{n \times n} = Cov(v)$ and $Q_{n_d} \in \mathbb{R}^{n_d \times n_d} = Cov(w)$, and for the measurement noise $R_v \in \mathbb{R}^{p \times p}$. Then define:

$$\tilde{Q} \triangleq \begin{bmatrix} Q_n & 0_{n \times n_d} \\ 0_{n_d \times n} & Q_{n_d} \end{bmatrix} \geq 0 \tag{46}$$

From $\tilde{A}, \tilde{B}, \tilde{C}, \tilde{Q}, R_v$ above, the matrix $P \in \mathbb{R}^{(n+n_d) \times (n+n_d)}$, $(P \geq 0)$ can be obtained from the following Discrete Algebraic Riccati Equation (DARE) as follows:

$$P = \tilde{A}P\tilde{A}^T + \tilde{Q} - \tilde{A}P\tilde{C}^T(\tilde{C}P\tilde{C}^T + R_v)^{-1}\tilde{C}P\tilde{A}^T \tag{47}$$

Once P is solved, the observer gains $L = [K^x \; K^w]^T$, $L \in \mathbb{R}^{(n+n_d) \times p}$ are obtained by:

$$L = \tilde{A}P\tilde{C}^T(\tilde{C}P\tilde{C}^T + R_v)^{-1} \tag{48}$$

The first n rows of L are K^x and the remaining n_d rows are K^w. After the state and disturbance estimates are obtained, $\hat{X}$ is fed back into the closed loop MPC formulation (akin to state feedback), and $\hat{w}$ is used to offset the setpoint (or reference signal) as:

$$\mathcal{R}_c = \mathcal{R} - \hat{w}_k \tag{49}$$

where $\mathcal{R}_c$ is the corrected setpoint, and in (49), since the disturbance is assumed to be the same for all states, the r best estimates of $\hat{w}_k$ can be chosen. In practicality, this implies that the observer can be defined beforehand and then implemented within the control system on real hardware.

2.7 MPC Stability

It is critical to design a closed loop nominal (default) MPC controller that is stable. Based on the literature, the traditional MPC algorithms calculate the predicted outputs Y_{pr} rather than the setpoints, from the state estimate $\hat{X}$. However, to further save computational time for the optimization, the predicted CVs Z_{pr} have been used instead. This reduces the dimension of the F, G, F_z, G_z matrices of the internal MPC model as usually, $r \leq p$ (if $m \leq p$) i.e the number of state variables to be controlled is less than the number of sensor outputs. Using this simplification, it was demonstrated in this study that a stable MPC controller can still be designed as long as the prediction and control horizons (N_p, N_c) are tuned properly. Sections 2.7.1 and 2.7.2 provides an example. In this case, the control horizon was set to $N_c = 1$ for both methods. The open loop systems are very similar however they do not have the exact same poles. This was done to demonstrate that regardless of the open loop system characteristics, both methods can yield closed loop stable controllers. Note that the pole zero plot presented is in the z plane. Stable poles lie within the unit circle ($|z| < 1$).

2.7.1 Efficient MPC Stability

As the efficient MPC does not place prediction points at each time step k from $k = t \rightarrow k = t + N_p$, it is more important to tune N_p even more carefully, especially in the case of single point prediction used in this study. Fig. 2.4 and Table 2.2 displays the closed loop system poles vs the same open loop model for different prediction horizons, $N_p = 1.0s, 3.0s, 5.0s$ respectively. It can be seen that as N_p increases, the overshoot of the closed loop poles decreases until a point where increasing N_p may increase the resulting overshoot. The settling time tends to increase due to the larger horizon length. Also making N_p too large in practicality will cause the MPC controller to miss a reference

signal if it changes within the horizon.

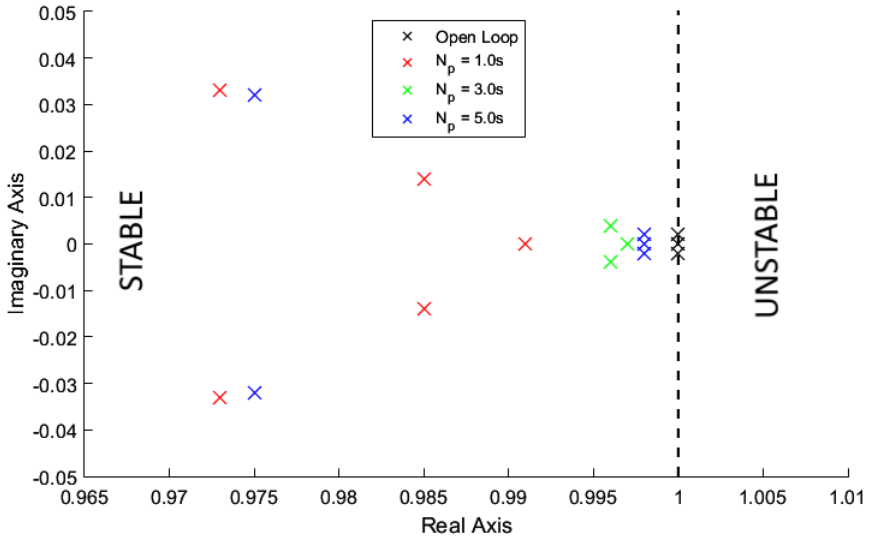

Figure 2.3: Pole Zero Plot - Efficient MPC Open vs. Closed Loop

Table 2.2: Open Loop vs. Closed Loop Characteristics for different Prediction Horizons - eMPC

System	**Eigenvalue**	**Settling Time** (s)	$\%$**Overshoot**
Open Loop	$0.975 \pm 0.032i$ $1 \pm 0.002i$ 1.00	1.64 4.01 2.626	9.55 79.55 0.0
$N_p : 1.0s$	$0.977 \pm 0.033i$ $0.985 \pm 0.014i$ 0.991	1.64 2.626 4.01	10.5 3.203 0.0
$N_p : 3.0s$	$0.975 \pm 0.032i$ 0.997 $0.996 \pm 0.004i$	1.64 11.60 10.93	9.55 0.0 4.89
$N_p : 5.0s$	$0.975 \pm 0.032i$ 0.998 $0.998 \pm 0.002i$	1.64 18.978 19.653	9.55 0.0 6.38

2.7.2 Integral Action MPC Stability

As the integral action MPC algorithm is very similar to the standard MPC algorithm where prediction points are placed at each time step ΔT, it is easier to tune N_p to design a closed loop stable controller. Here, $5, 10$ and 30 prediction points are used. Setting N_p to a small number of points does not affect the performance much (except stabilizing the system). However, setting the number of prediction points to a large enough value (e.g. 30), drastically improves stability and decreases

the pole overshoot (Fig. Y and Table 2.3). Another major advantage is that increasing N_p to a large value also brings the poles close to zero, thus also drastically decreasing the settling time as well. It can be concluded that the integral action MPC method is far superior in terms of closed loop stability and performance. Granted however, both methods do yield closed loop stable controllers.

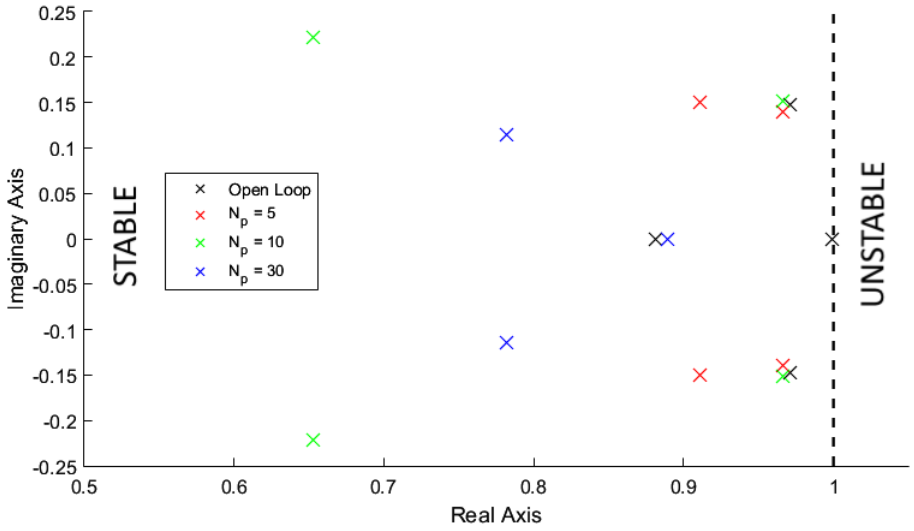

Figure 2.4: Pole Zero Plot - Integral Action MPC Open vs. Closed Loop

Table 2.3: Open Loop vs. Closed Loop Characteristics for different Prediction Horizons - iMPC

System	**Eigenvalue**	**Settling Time** (s)	$\%$**Overshoot**
Open Loop	$0.971 \pm 0.147i$ 0.881 0.999	2.493 8.44 8.44	68.30 0.0 0.0
$N_p : 5$	$0.911 \pm 0.15i$ $0.966 \pm 0.14i$	2.49 8.44	21.2 59.65
$N_p : 10$	$0.966 \pm 0.151i$ $0.653 \pm 0.222i$	9.087 0.538	64.069 2.829
$N_p : 30$	$0.782 \pm 0.115i$ 0.021 0.889	0.85 0.052 1.697	0.64 0.0 0.0

2.8 The MPC Algorithm

Any MPC based feedback controller is implemented as follows, and is represented by Algorithm 1 and in block diagram form, by Fig. 2.5 [21]. Note that the predicted CVs in this algorithm are

calculated by F, G, this is just a change of notation. The total simulation time is given by T, and the system matrices are given by A, B, C, A_c, B_c. Based on which specific MPC method is used, either the continuous time models are needed or they may not be. The unconstrained feedback is given by K and the observer gain is given by L.

Algorithm 1 Generic MPC Algorithm

Require: $A, B, C, F, G, \Delta T, L, K$
Require: A_c, B_c, N_p, N_c, T, M
Ensure: $\Delta T > 0, T > 0, \exists[\hat{X}(0), U(0), X(0), Y(0)]$
$k = 0$
$k_f = T/\Delta T$
Ensure: $N_p > 0$ and $N_c \leq N_p$
for $k = 0 \rightarrow k_f$ **do** ▷ k is the time step
$X_{k+1} = AX_k + BU_k + w_k$
$Y_{k+1} = CX_{k+1} + v_k$
Obtain $\hat{X} \leftarrow$ from Y, C, L ▷ Call the observer
$Z_{pr} = F\hat{X}_k + GU_k$ ▷ Predicted Outputs
Obtain $E \leftarrow$ from Z_{pr} ▷ (Error)
$U_u = KE$ ▷ Unconstrained Input
Obtain $\gamma_k \leftarrow$ from U_k
    if $MU_u \leq \gamma_k$ **then** ▷ Does U_u meet constraints?
        $U = U_u$
        GOTO 17
    else Obtain U_c from Algorithm 3
        $U = U_c$
    end if
$k = k + 1$
end for

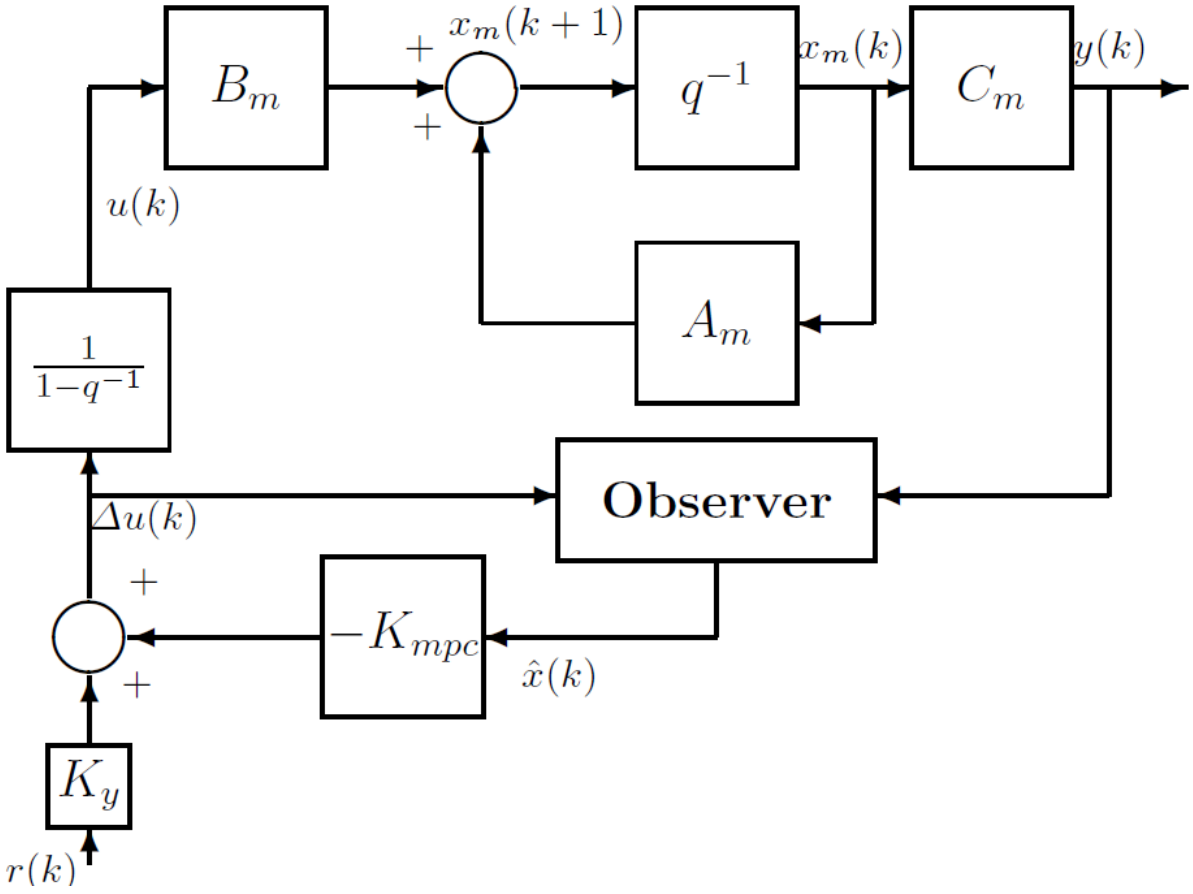

Figure 2.5: Model Predictive Control - Block Diagram

Chapter 3

Fault Detection and Diagnosis (FDD)

3.1 Uncertain Systems

Before proceeding with the proposed fault detection and estimation method, it is crucial to gain a theoretical understanding of the various methods used. Firstly, an "uncertain" dynamical system is any type of system subject to time varying uncertainties in its behaviour. Notice that in 2.6, as the disturbance is assumed to be constant, this type of system is not considered uncertain. There has been significant research done in observer design for these systems. The standard Kalman Filter consists of "Predict" and "Update" components [23] and is illustrated here (Initial guess is $[.]_0$):

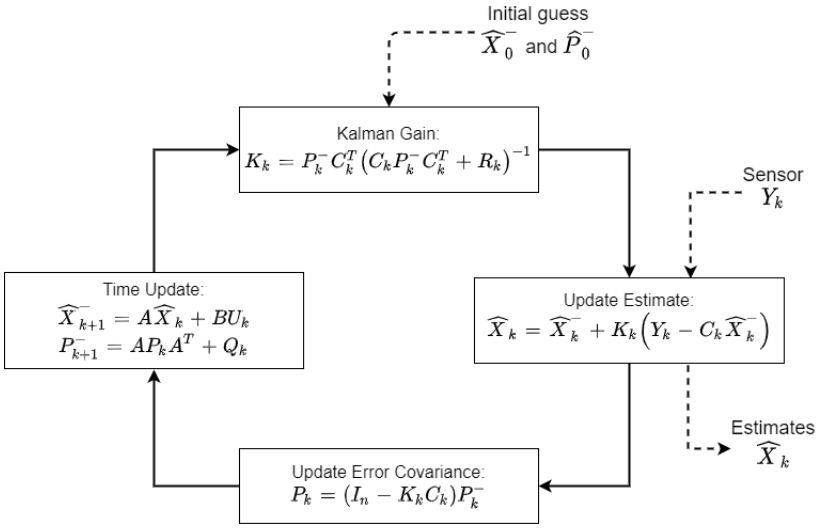

Figure 3.1: Kalman Filter Loop

Looking at the general Kalman filter, an initial guess of the error covariance P is required. For the system to be stable, the term $A - KC$ must be stable. Clearly, the general Kalman Filter is not the same as the steady state observer (or steady state Kalman Filter) because in that case, the Kalman gain is optimally solved for beforehand and stays constant. Generally, the term "observer" is used for estimators with a steady state (or constant) gain. However, if the observer is designed using adaptive or sliding mode laws, then this is termed an "adaptive observer". Though not identical to the Kalman Filter, it still serves the same purpose.

Moving on to fault detection in linear systems, since the Loss Of Effectiveness parameter f is not measured directly, the objective is to estimate $\hat{f}$ in the event of LOE occurrence. Many methods have been researched to figure out the best way to solve for $\hat{f}$ for linear systems. The Two Stage Kalman filter has been widely used as per [24], [25] and [26]. However, this method contains a lot of numerical operations including coupling and bias free estimation equations, and also requires matrix inversion, which is not ideal. On the contrary, classical observer based methods are more effective as they provide both state and fault estimates with minimal computations.

When it comes to diagnosing and detecting faults in linear or nonlinear systems, the concept of discontinuous observers has been used and an extended discussion is provided in [27]. These discontinuous observers are based on sliding mode theory, and are used to design controllers for uncertain dynamical systems. Speaking in terms of fault detection, any system in the presence of unknown faults (either in the system or actuators) is considered uncertain (due to the unknown effective input in relation to the actual input). The task of this observer is to estimate this effective input. A Lyapunov based method is used to design these observers and it provides an accurate estimate of the error (or fault) within the uncertain system. One condition to keep in mind is that for this observer to work, the error or uncertainty must be bounded. This is inherently true for a faulty actuator because the fault parameter is bounded, as explained in 2.2. The "Walcott and Zak" observer is a type of Lyapunov based sliding mode observer which is very useful for estimation of uncertainty and is highlighted in [27], along with more information provided by the original authors in [28]. In general, consider any dynamic system $\mathcal{G}$ containing an unknown error ξ given by:

$$\mathcal{G}(\xi) = f(t, X, U) \longrightarrow |\xi| \leq \rho, \ \forall X \in \mathbb{R}^n \tag{50}$$

where once again, the states are X, inputs are U and the fault error is ξ, and as expressed in (50), ξ is bounded between ρ. The "Walcott and Zak" observer considers problems of this nature to provide estimates of states in the presence of faults (and the fault parameter ξ as well).

3.2 State and Fault Parameter Estimation

The method used for state and fault estimation is taken from [29] and it is based on the "Walcott and Zak" observer. One of the primary benefits is that the required parameters can be defined ahead of time. This allows rapid real time detection of the effective inputs and estimation of $\hat{f}$. Firstly, the Lyapunov equation in discrete time is:

$$A^T P A - P + \tilde{Q} = 0 \tag{51}$$

where $P, Q \in \mathbb{R}^{n \times n}$ and if $P > 0,\ Q > 0$, then A is discrete-time stable and all state trajectories $X_{k+1} = AX_k$ are bounded. Lyapunov theory has been used to design asymptotically stable observers. Next, let $L \in \mathbb{R}^{n \times p}$ represent the observer gain i.e. then the closed loop observer matrix is $A_{cl} = A - LC$. In other words A can be unstable, however A_{cl} must be Hurwitz. A bank of observers can then be designed provided the following Observer Existence Assumption (OEA) holds true (in discrete time).

$$A_{cl}^T P A_{cl} - P + \tilde{Q} = 0 \tag{52}$$

$$DC = (PB)^T \tag{53}$$

Another condition to design the bank of observer is that the nominal input matrix B must be of full column rank. This indicates that CB must also be full column rank. To begin, let $U_f \in \mathbb{R}^m$ represent the effective input vector. In other words, if $f_1 = 0.75$, then $U_{f1} = 0.25U_1$. Considering that generally only the noisy sensor outputs Y_k are available, The following (discrete time equivalent) expressions solve for $\hat{X}$ and $\hat{U}_f$.

$$\hat{X}_{i(k+1)} = A\hat{X}_{ik} - L(C\hat{X}_{ik} - Y_k) + b_i\hat{U}_{fi} + \sum_{j=1, j\neq i}^{m} b_j U_j \tag{54}$$

$$\hat{U}_{fi} = \hat{U}_{fi} - 2\epsilon_i(C\hat{X}_{ik} - Y_k)^T d_i \tag{55}$$

where $1 \leq i \leq m$ and $1 \leq j \leq m$. Looking at (54) and (55), it is clear that they have to be repeated m times for m inputs. This indicates that there are m sets of $\hat{X}$ and any one of them can be used for feedback. As a result for each effective input U_{fi}, the parameter ϵ_i $(\forall i)$ must be carefully tuned to provide accurate results for each estimated $\hat{X}$. The proof behind the derivation of (54) and (55) is provided within [29].

In (54) the term b_i is the i^{th} column of B, and L is the observer gain obtained from the DARE (47). It is given by:

$$L \triangleq A\tilde{P}C(C\tilde{P}C^T + \tilde{R})^{-1} \tag{56}$$

Here, $\tilde{P} \in \mathbb{R}^{p\times p}$ is the solution to the DARE, $\tilde{R} \in \mathbb{R}^{p\times p}$ is a postive definite matrix defining the noise covariance $Cov(v_k)$. In (55) the term d_i is the i^{th} column of D and the matrix $D \in \mathbb{R}^{m\times n}$ is obtained from (53), and $P \in \mathbb{R}^{n\times n}$ is obtained from solving (51) using A_{cl} and a positive definite matrix $\tilde{Q}$.

3.2.1 Loss Of Effectiveness Estimation

With the observer, one can have simultaneous values of U and U_f. These are passed through a standard moving average filter having window length $M_v \in \mathbb{R}_+$. Each LOE estimate is then calculated as follows $\forall i$: (Note $1 \leq i \leq m$).

$$\hat{f}_i = max\Big[\frac{\Gamma(U_f)}{\Gamma(U)}\Big] - 1 \tag{57}$$

where Γ is the Fast Fourier Transform (FFT) operator, also taken over a moving window $M_{v2} \in \mathbb{R}_+$. The FFT method is one of the most accurate ways to estimate the ratio between two noisy signals. Fig. 3.2 provides a schematic representation.

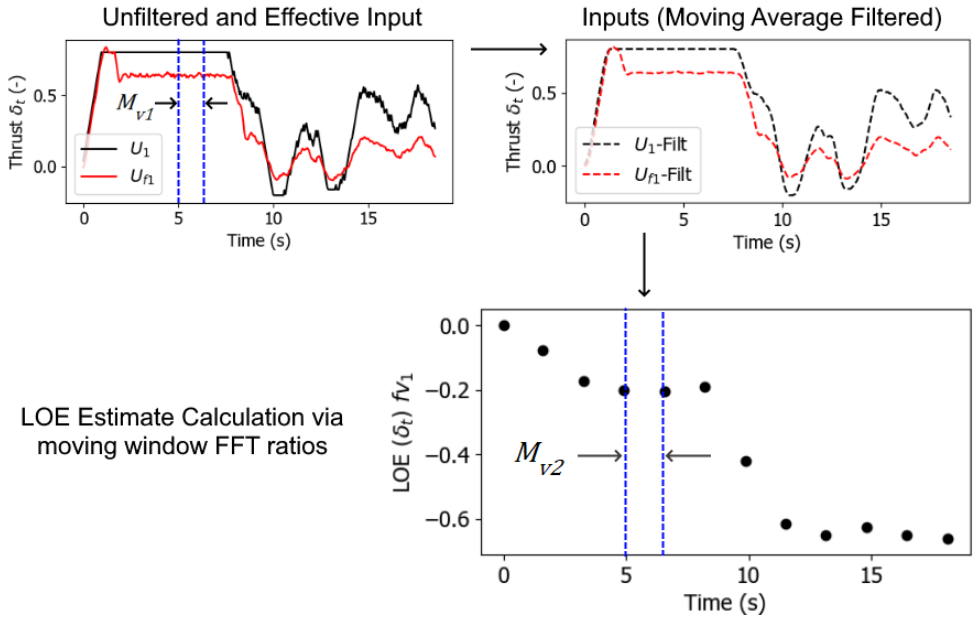

Figure 3.2: Fault Estimation via Moving Window FFT

3.3 Using FDD within MPC

Another key contribution of this study was integrating these FD algorithms within the two MPC controllers via simultaneous state and fault estimation. Previously, the majorty of MPC research work was done for chemical engineering applications [8] and for low complexity SISO systems such as quadrotors [24]. The large dimensionality of the MIMO system models for fixed wing aircraft is a major drawback of implementing FDD based MPC controllers for this specific application. For the Integral Action MPC, as the optimization variable is ΔU, it already has a degree of implicit fault tolerance capability. Hence, $\hat{X}$ can be used for feedback, and no active reconfiguration method is needed. As the efficient MPC has a greater risk of destabilizing in the presence of faults due to solving U directly, reconfiguration is required here. It is applied to the unconstrained solution from (17):

$$U = KE + \underbrace{(U - \hat{U}_f)}_{Reconfiguration} \tag{58}$$

The reconfiguration must be done for the unconstrained solution because regardless if faults are present, the overall actuator movement must still not exceed its position and rate constraints. This

indicates that if the fault is large, the MPC based feedback controller will eventually become unstable, that is also the case without any reconfiguration mechanism. It is also important to emphasize that the LOE faults considered do not affect the ability of the actuator's motion i.e it does not reduce or increase their constraints. The LOE affects the actuator's performance rather. For instance, consider the spoiler actuator on an aircraft. If the spoiler is deployed to it's maximum position, any wing damage will reduce it's generated drag force which will affect how "effective" the spoiler is. This is what actuator LOE takes into account.

3.4 Comparison to Previous Methods

As mentioned in 3.1, the Two stage Kalman filter used to detect the LOE in an actuator contains a lot of operations in comparison to the more simpler observer based method presented in this study. Fig. 3.3 presents a comparison of the two methods. It can be clearly seen that the observer based method contains a maximum of $4 \cdot m$ operations, while the two stage Kalman Filter has a fixed number of 18 operations. Hence, for the majority of LTI systems which do not have many inputs, the observer method is superior. Furthermore, several of these equations within the TSKF contain matrix inversion unlike the observer method. The TSKF method however is superior for Linear Time Varying (LTV) systems since it is able to calculate $\hat{f}$ as it contains "Predict" and "Update" (corrector) equations which automatically calculate the required observer gain in real time. This indicates that it can handle stochastic systems. The observer method is better for more determinstic systems such as LTI models where although the fault can vary with time, the system parameters stay constant. Additionally, the TSKF algorithm does not require a moving window FFT algorithm to divide the effective and actual inputs. In conclusion, for this study it is superior to use the observer method as only LTI systems are considered and it is significantly faster, making it ideal for systems with limited computational complexity.

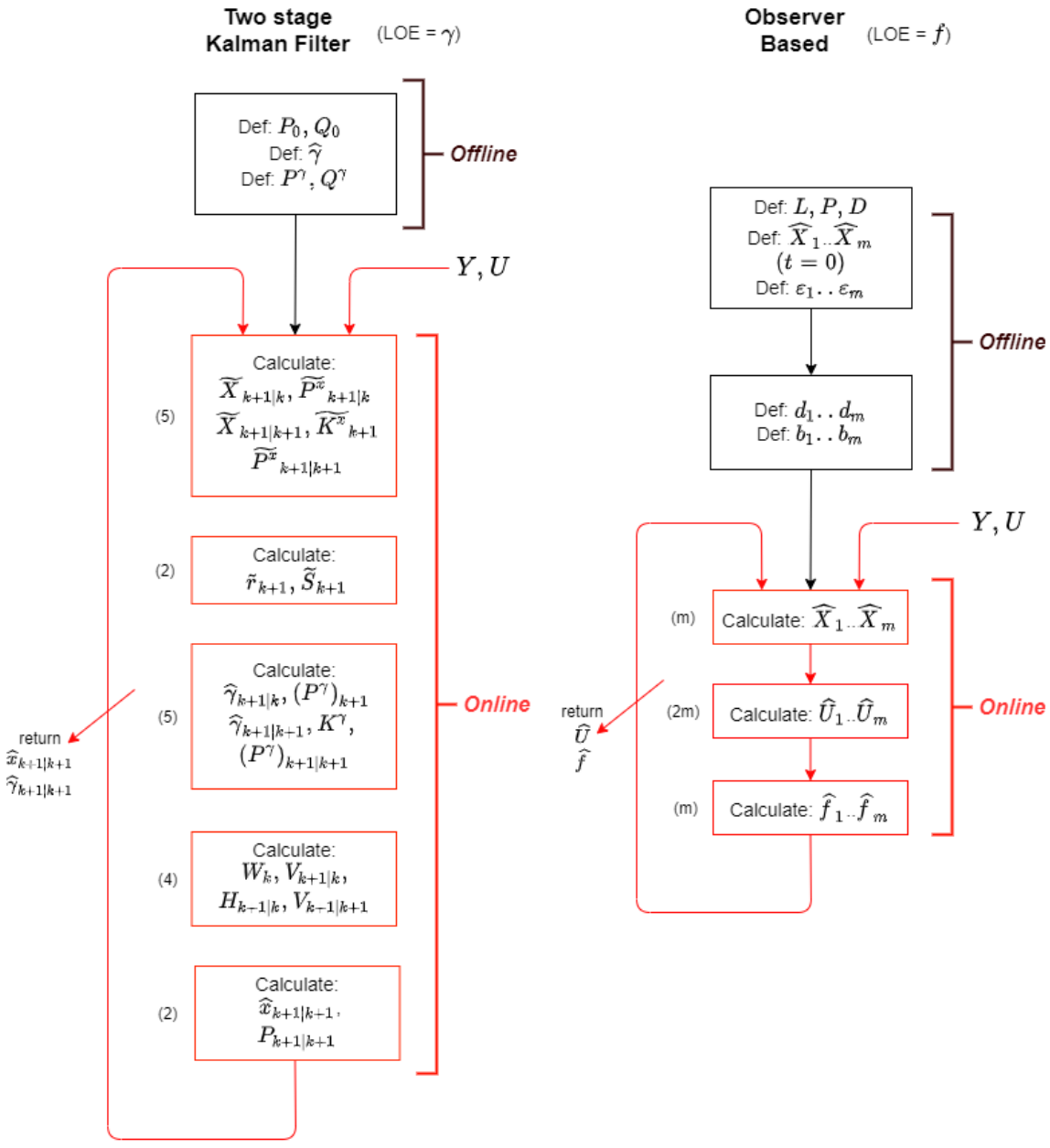

Figure 3.3: Two stage Kalman filter vs Observer

Chapter 4

Aircraft Mathematical Model

4.1 Fixed-Wing Aircraft

A contribution of this study was the novel application of the above methods to fixed-wing aircraft. Aircraft contain multiple inputs such as thrust, elevator, rudder, aileron, and flaps along with multiple outputs such as the six-degree-of-freedom (6-DOF) orientation: NED (North-East-Down) position, roll/pitch/yaw angles, along with velocities and accelerations. In addition, each output will intuitively have a different response time with respect to each input. Hence, this makes it ideal to use MPC techniques for this application. However, the onboard flight computer's available capacity must be taken into consideration before choosing the ideal MPC algorithm to use. This is the underlying reason why the efficient MPC algorithm was proposed by the authors in [10].

4.1.1 Nonlinear Model

Assuming the aircraft is modeled as a rigid body, the flat Earth 6-DOF equations of motion are as follows. Note: $s\theta = sin(\theta),\ c\phi = cos(\phi),\ t\psi = tan(\psi)$:

$$X - mgs\theta = m(\dot{u} + qw - rv) \tag{59}$$

$$Y + mgc\theta s\phi = m(\dot{v} + ru - pw) \tag{60}$$

$$Z + mgc\theta c\phi = m(\dot{w} + pv - qu) \tag{61}$$

$$L = I_x\dot{p} - I_{xz}\dot{r} + qr(I_z - I_y) - I_{xz}pq \tag{62}$$

$$M = I_y\dot{q} + rp(I_x - I_z) + I_{xz}(p^2 - r^2) \tag{63}$$

$$N = -I_{xz}\dot{p} + I_z\dot{r} + pq(I_y - I_x) + I_{xz}qr \tag{64}$$

$$p = \dot{\phi} - \dot{\psi}s\theta \tag{65}$$

$$q = \dot{\theta}c\phi + \dot{\psi}c\theta s\phi \tag{66}$$

$$r = \dot{\psi}c\theta c\phi - \dot{\theta}s\phi \tag{67}$$

$$\dot{\theta} = qc\phi - rs\phi \tag{68}$$

$$\dot{\phi} = p + t\theta(qs\phi + rc\phi) \tag{69}$$

$$\dot{\psi} = \frac{qs\phi + rc\phi}{c\theta} \tag{70}$$

$$\begin{bmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \\ \frac{dz}{dt} \end{bmatrix} = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{71}$$

Table 4.1 contains the definitions of the terms in (59) to (71).

Table 4.1: Aircraft Equations of Motion - Terms and Definitions

Term	**Definition**	**Units**
$X,\ Y,\ Z$	Body Axis Forces in X, Y and Z direction	Newtons (N)
$L,\ M,\ N$	Body Axis Moments in X, Y and Z direction	Newton-Meter (Nm)
$p,\ q,\ r$	Body Axis Angular Velocities in X, Y and Z direction	Radians/sec (rad/s)
$u,\ v,\ w$	Body Axis Linear Velocities in X, Y and Z direction	Meters/sec (m/s)
$\theta,\ \phi,\ \psi$	Orientation (Roll, Pitch, Yaw) Angles	Radians (rad)
$x,\ y,\ z$	Location in 3D space with respect to Inertial Axis	Meters (m)
I_x, I_y, I_z, I_{xz}	Moments of Inertia with respect to Body X, Y, Z, and XZ (cross) axes	Kilogram-sq.-Meter (kgm^2)

From (59) to (71), further parameters characterizing the motion of an air vehicle can be calculaed and used for control system design and simulation. These are for the angle of attack (α), flight velocity (V_T), flight path angle (γ), altitude (h), and sideslip angle (β):

$$\alpha = tan^{-1}\Big(\frac{w}{u}\Big) \tag{72}$$

$$\gamma = \theta - \alpha \tag{73}$$

$$V_T = \sqrt{u^2 + v^2 + w^2} \tag{74}$$

$$h = -z \tag{75}$$

$$\beta = sin^{-1}\Big(\frac{v}{V_T}\Big) \tag{76}$$

4.1.2 Linearized Model and Inputs

As the nonlinear model is highly complex, the linearization process is done at a specific flight condition. For instance, possible linearization points are:

(1) Straight and Level Flight

(2) Steady Pitch up / down

(3) Steady Turn (Roll)

(4) Steady Heading (Yaw)

In the majority of situations, straight and level flight is chosen to be the linearization point because it is a good starting point to design controllers for. Furthermore, a small angle approximation can be used for designing pitch, roll or yaw controllers, or steady climb/ descent. For the straight and level flight condition, the following equilibrium points are used:

- $v = w = 0, p = q = r = 0$
- $\dot{\theta} = \dot{\phi} = \dot{\psi} = 0, \theta = \phi = \psi = 0$
- $\frac{dy}{dt} = \frac{dz}{dt} = 0$
- $L = M = N = 0$

Table 4.2: Aircraft Control Surface Inputs

Term	Definition	Units
δ_t	Thrust Setting	$(-)$
δ_e	Elevator Deflection	rad
δ_a	Aileron Deflection	rad
δ_r	Rudder Deflection	rad
δ_c	Canard Deflection	rad
δ_f	Flaperon Deflection	rad
δ_{eR}, δ_{eL}	Right / Left Horizontal Tail Deflection	rad
δ_{fR}, δ_{fL}	Right / Left Flaperon Deflection	rad

The input actuators for the aircraft are represented in Table 4.2. Once linearization is performed in the chosen flight condition, the dynamics are decoupled into longitudinal and lateral modes, which makes it a lot simpler to design linear controllers. All inputs may not necessarily be present for all types of aircraft.

4.1.3 Longitudinal Dynamics

The longitudinal motion of the aircraft only considers pitching motion, and movement in the 2D (XZ) plane. As a result, this is a three degree of freedom (3-DOF) planar model. A schematic is provided in Fig. 4.1. The states and control inputs are as follows [19].

$$X = [h\ \theta\ v\ \alpha\ q\ \delta_t\ \delta_e]^T \tag{77}$$

$$Y = [h\ \theta\ v\ \alpha\ q]^T \tag{78}$$

$$U = [\delta_t\ \delta_e]^T \tag{79}$$

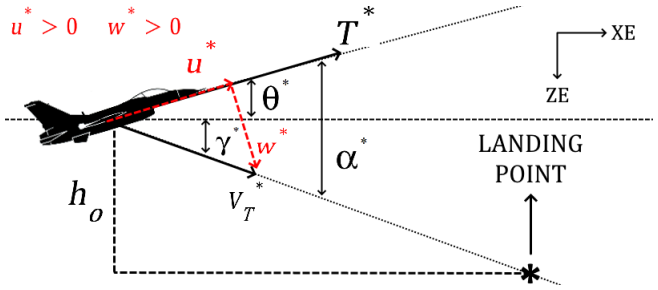

Figure 4.1: Longitudinal Dynamics - Free Body Diagram

4.1.4 Lateral Dynamics

The lateral motion of the aircraft only considers pitching motion, and movement in the XY plane. Hence this is also a 3-DOF planar model. A schematic is provided in Fig. 4.2. The states and control inputs are as follows:

$$X = [\phi\ \psi\ v\ \beta\ p\ r\ \delta_a\ \delta_r]^T \tag{80}$$

$$Y = [\phi\ \psi\ v\ \beta\ p\ r]^T \tag{81}$$

$$U = [\delta_a\ \delta_r]^T \tag{82}$$

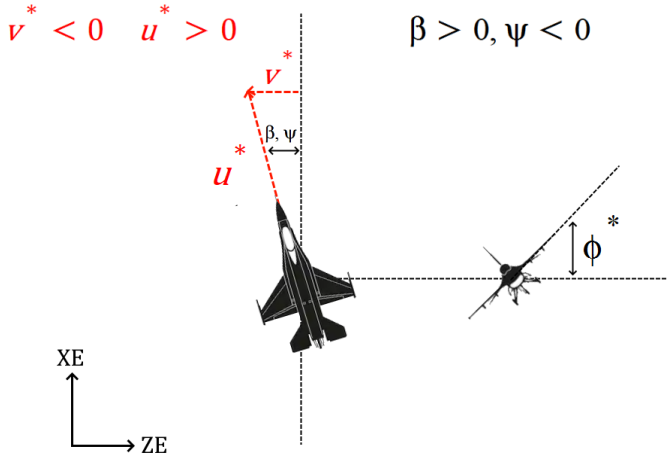

Figure 4.2: Lateral Dynamics - Free Body Diagram

4.1.5 Coupled Longitudinal and Lateral Dynamics

Sometimes, a linearization is performed however the longitudinal and lateral dynamics are not coupled (i.e. within the states or the inputs). This is the case in [30]. Hence, this is a six-degree of freedom (6-DOF) model and its state space representation can be given as:

$$X = [\theta\ u\ \alpha\ q\ \phi\ \beta\ p\ r]^T \tag{83}$$

$$Y = X \tag{84}$$

$$U = [\delta_{eR}\ \delta_{eL}\ \delta_{fR}\ \delta_{fL}\ \delta_c\ \delta_r]^T \tag{85}$$

4.2 System Models Used

4.2.1 F-16 Aircraft

For the MPC design, an open source fixed wing flight dynamics model was used, and it is taken from [31], and can also be downloaded from this link. This is a full Matlab/Simulink based high fidelity nonlinear flight dynamics model for the Lockheed Martin F-16 Fighting Falcon, one of the greatest aircraft ever built! The LINMOD command in Matlab/Simulink is used to obtain the linearized longitudinal and lateral models and then controller design can be performed. The equilibrium flight condition and setting can be chosen by the user. More information about the LINMOD command can be found at this link.

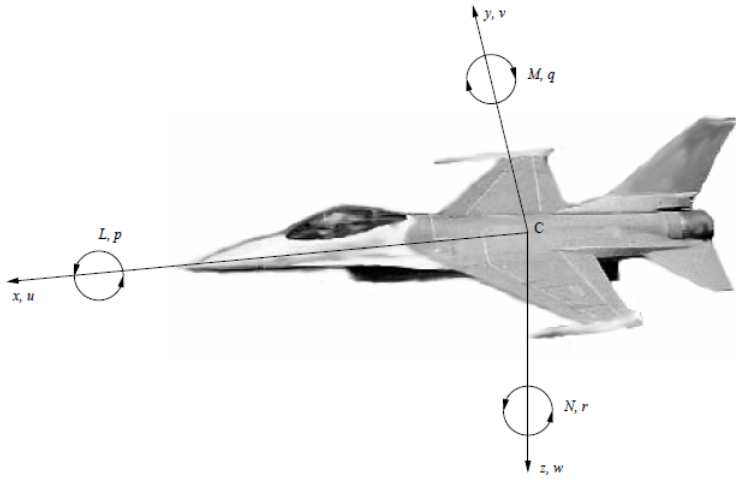

Figure 4.3: F-16 Aircraft

4.2.2 F-16 Longitudinal Model

The continuous time longitudinal model of the F-16 is provided. To design the MPC controllers, the model was first discretized using a Zero Order Hold. Here are the continuous time parameters. This model is open loop marginally stable, and this can be demonstrated by the pole-zero plot in Fig. 4.4.

$$A_c = \begin{bmatrix} 0 & 500 & 0 & -500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0.0001 & -32.2 & -0.013 & -2.948 & -1.028 & 0.002 & 0.102 \\ 0 & 0 & -0.0003 & -0.751 & 0.928 & 0 & -0.002 \\ 0 & 0 & 0 & -1.837 & -1.027 & 0 & -0.134 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -20.2 \end{bmatrix} \tag{86}$$

$$B_c = \begin{bmatrix} 0^{5\times 2} \\ 1 & 0 \\ 0 & 20.2 \end{bmatrix} \tag{87}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -7 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{88}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{89}$$

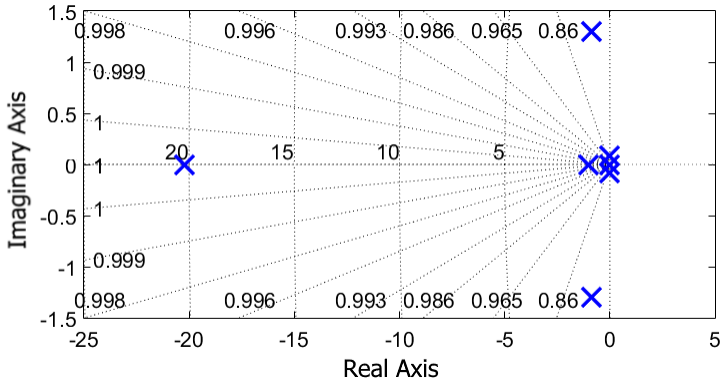

Figure 4.4: Longitudinal Dynamics - Pole Zero Plot

4.2.3 F-16 Lateral Model

Here is the continuous time F-16 lateral model. Once again, the model was first discretized using a Zero-Order-Hold. This model is open loop marginally stable, demonstrated in Fig. 4.5. The poles in continuous time are $0.00, -0.35 \pm 2.88i, -0.012, -2.55, -0.013, -1.00, -20.20$ and -20.20.

$$A_c = \begin{bmatrix} 0 & 0 & 0 & 0 & 1.00 & 0.06 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.00 & 0 & 0 \\ 0 & 0 & -0.01 & 0 & 0 & 0 & 0 & 0 \\ 0.06 & 0 & 0 & -0.24 & 0.06 & 0.99 & 2e-3 & 0.01 \\ 0 & 0 & 0 & -25.23 & -2.67 & 0.58 & -0.54 & 0.07 \\ 0 & 0 & 0 & 7.01 & -0.04 & -0.37 & -0.03 & -0.05 \\ 0 & 0 & 0 & 0 & 0 & 0 & -20.2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -20.2 \end{bmatrix} \tag{90}$$

$$B_c = \begin{bmatrix} 0^{5\times 2} \\ 20.2 & 0 \\ 0 & 20.2 \end{bmatrix} \tag{91}$$

$$C = [I^6 \; 0^6 \; 0^6] \tag{92}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{93}$$

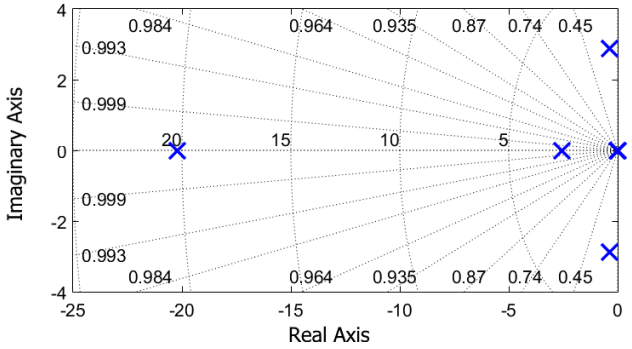

Figure 4.5: Lateral Dynamics - Pole Zero Plot

4.2.4 F-16 Coupled Longitudinal and Lateral Model

Another simulation was carried out with coupled linearized longitudinal and lateral dynamics to study how the efficient MPC would perform. The Advanced Fighter Technology Integration (AFTI) F-16 was a combined NASA and Department of Defense (DoD) program to test possible future aircraft systems. Its model is taken from [30]. This model is open loop unstable and its pole zero plot is presented in Fig. 4.6. The poles in continuous time are: -3.22, $-0.0047 \pm 0.05i$, 0.9642, $-0.39 \pm 2.96i$, -0.027 and -2.69.

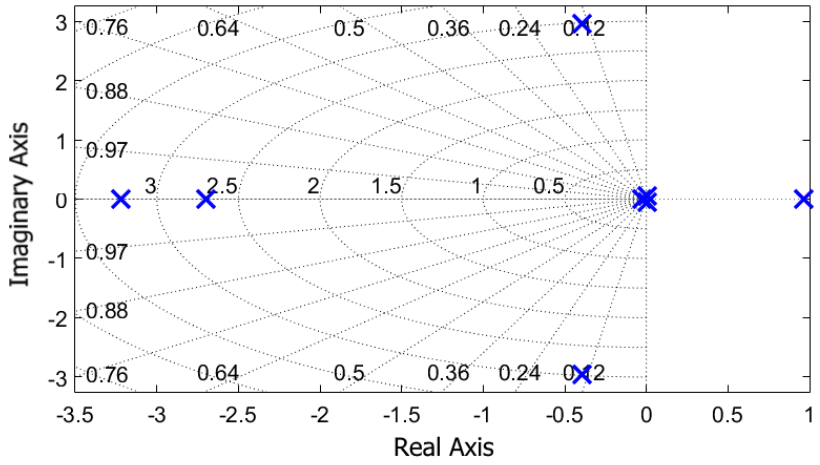

Figure 4.6: Longitudinal and Lateral Dynamics - Pole Zero Plot

$$A_c = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ -32.183 & 0.0121 & 38.2906 & -30.1376 & 0 & 0 & 0 & 0 \\ -0.00112 & -0.00002 & -1.4845 & 0.9948 & 0 & 0 & 0 & 0 \\ -0.0003 & -0.00013 & 4.2717 & -0.7772 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0.0345 & -0.3435 & 0.0326 & -0.9976 \\ 0 & 0 & 0 & 0 & 0 & -55.253 & -2.8000 & 0.1457 \\ 0 & 0 & 0 & 0 & 0 & 7.237 & -0.0232 & -0.3625 \end{bmatrix} \tag{94}$$

$$B_c = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1.003 & 1.003 & 1.1584 & 1.1584 & 0 & 0 \\ -0.0746 & -0.0746 & -0.1224 & -0.1224 & 0 & 0 \\ -12.029 & -12.029 & -3.2363 & -3.2363 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0.0133 & -0.0133 & -0.0007 & 0.0007 & 0.0267 & 0.0370 \\ -25.3645 & 25.3645 & -25.5251 & 25.5251 & 5.5319 & 10.3955 \\ -2.5686 & 2.5686 & -0.6250 & 0.6250 & 5.8925 & -5.8089 \end{bmatrix} \tag{95}$$

$$C = [I^8] \tag{96}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{97}$$

4.2.5 Generic Longitudinal Model

For the simulations demonstrating the fault tolerant control shown later in 5.2.2, this generic longitudinal model of a fixed wing aircraft was used, and it is taken from [10].

$$A_c = \begin{bmatrix} -0.026 & 0.074 & -0.804 & -9.809 & 0.000 \\ -0.242 & -2.017 & 73.297 & -0.105 & -0.001 \\ 0.003 & -0.153 & -2.941 & 0.000 & 0.000 \\ 0.000 & 0.000 & 1.000 & 0.000 & 0.000 \\ -0.011 & 1.000 & 0.000 & -75.000 & 0.000 \end{bmatrix} \tag{98}$$

$$B_c = \begin{bmatrix} 4.594 & 0.000 \\ -0.0004 & -13.735 \\ 0.0002 & -24.410 \\ 0^{2\times 2} \end{bmatrix} \tag{99}$$

$$C = [I^5] \tag{100}$$

$$H = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{101}$$

This model is open loop marginally stable, demonstrated by the pole-zero plot in Fig. 4.7. The poles (continuous time) are $-2.48 \pm 3.32i$, $-0.0115 \pm 0.16i$, and -0.0007.

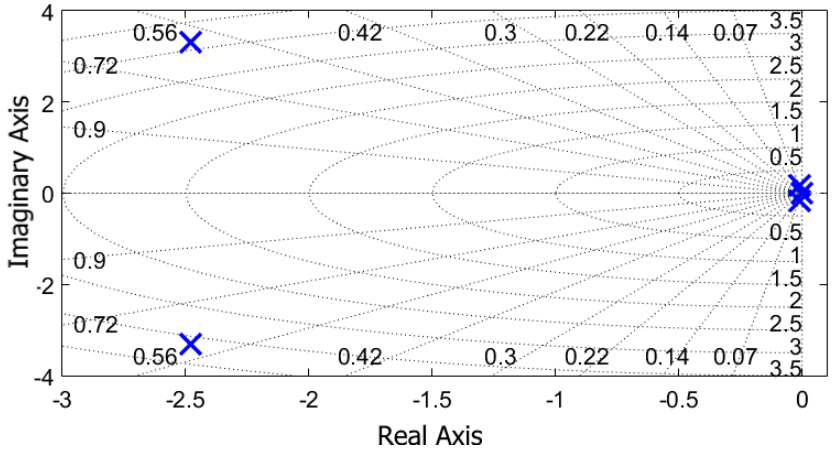

Figure 4.7: Generic Longitudinal Model - Pole Zero Plot

4.2.6 Reduced Order Longitudinal and Lateral Model

A final simulation was performed with reduced order longitudinal and lateral models of the F-16 obtained from [32]. The longitudinal states and system matrices in continuous time are as follows:

$$X_{lon} = Y_{lon} = [V_T \ \alpha \ \theta \ q]^T \tag{102}$$

$$U_{lon} = [\delta_T \ \delta_e]^T \tag{103}$$

$$A_{lon} = \begin{bmatrix} -0.0172 & -3.8858 & -32.1696 & -1.1096 \\ -0.0026 & -0.7506 & 0.00 & 0.9278 \\ 0.00 & 0.00 & 0.00 & 1.00 \\ -0.74e-12 & -4.2783 & 0.00 & -1.2612 \end{bmatrix} \tag{104}$$

$$B_{lon} = \begin{bmatrix} 19.722 & 0.09849 \\ -0.0026 & -0.0016 \\ 0.00 & 0.00 \\ 0.00 & -0.1386 \end{bmatrix} \tag{105}$$

$$C = [I^4] \tag{106}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{107}$$

The open loop longitudinal pole zero plot is shown in Fig. 4.8. As clearly seen it is unstable, with two complex poles outside the stable plane.

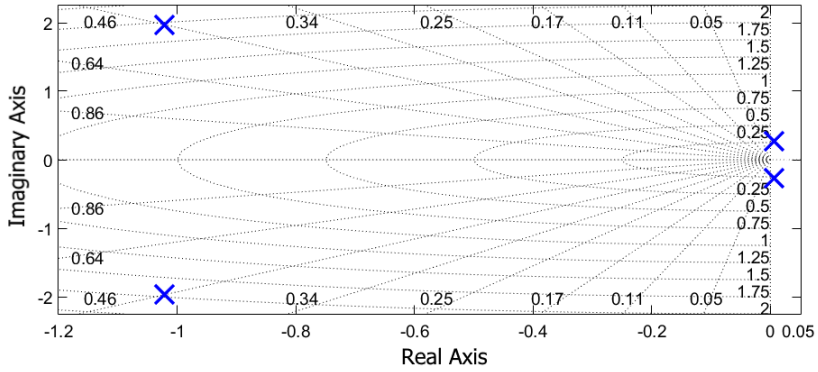

Figure 4.8: Reduced Order Longitudinal Model - Pole Zero Plot

Next, the lateral states and system matrices in continuous time are as follows:

$$X_{lat} = Y_{lat} = [\beta \ \phi \ p \ r]^T \tag{108}$$

$$U_{lat} = [\delta_a \ \delta_r]^T \tag{109}$$

$$A_{lat} = \begin{bmatrix} -0.2372 & 0.0642 & 0.0663 & -0.992 \\ 0.00 & 0.00 & 1.00 & 0.0662 \\ -25.533 & 0.00 & -2.6634 & 0.5906 \\ 7.692 & 0.00 & -0.0395 & -0.3851 \end{bmatrix} \tag{110}$$

$$B_{lat} = \begin{bmatrix} 0.00022 & 0.000593 \\ 0.0 & 0.0 \\ -0.5415 & 0.0938 \\ -0.0241 & -0.0489 \end{bmatrix} \tag{111}$$

$$C = [I^4] \tag{112}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{113}$$

The open loop lateral pole zero plot is shown in Fig. 4.9, and is marginally stable. The poles are: $-2.55, -0.008$ and $-0.40 \pm 2.65i$.

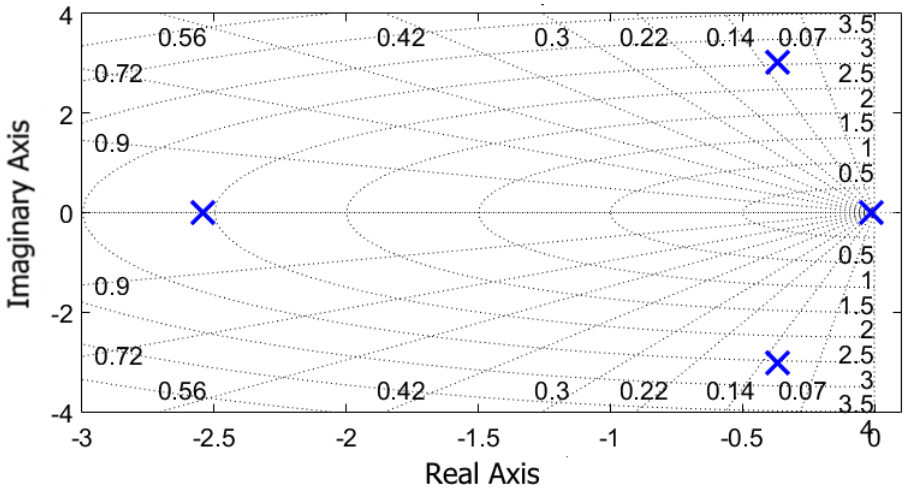

Figure 4.9: Reduced Order Lateral Model - Pole Zero Plot

Chapter 5

Simulation Studies

5.1 Fault-Free Conditions

The first two simulations were to test the performance of the novel constrained QP program formulated along with the disturbance observer in 5.1.1 and 5.1.2. The simulations were for the efficient MPC algorithm, and were performed in the Python programming language, and successful performance of the QP program along with the disturbance observer is achieved. The simulation in 5.1.1 is fault free and noise free as it just tests the QP program, and that in 5.1.2 contains process and measurement noise and a nonzero disturbance. No actuator fault was present for the time being.

5.1.1 Longitudinal Model

This is a simulation performed on the F-16 longitudinal model from 4.2.2. The control variables are h and v (incremental i.e Δh, Δv). The position and rate constraints on the actuators δ_t, δ_e are presented in Table 5.1. This simulation is taken from [19].

Table 5.1: F-16 Actuator Constraints

Actuator Type	**Constraints**	
	Absolute	***Incremental*** (Δ/s)
Thrust (δ_t^{a})	$0.0 \leq \delta_t \leq 1.0$	$-4.0 \leq \Delta(\delta_t) \leq 4.0$
Elevator (δ_e)	$-0.4363 \leq \delta_e \leq 0.4363$	$-1.04 \leq \Delta(\delta_e) \leq 1.04$

[a]Approximate Value

The parameters are presented in Table 5.2. Note that N_{p_1} corresponds to the horizon for h and N_{p_2} corresponds to the horizon for v. Recall that since the efficient MPC is used, this allows defining multiple horizons, one for each CV. Note: Refer to Appendix B for more information regarding the last two rows (Error convergence and PQP vector)

Table 5.2: Simulation Parameters - Longitudinal Model

Parameter	***Description***	***Value***
$\Delta T, T$	Sampling, Simulation Time	$0.01s, 60s$
N_{p_1}, N_{p_2}	Prediction Horizons	$1.5s,\ 2.0s$
h_0, v_0	Trim Height & Speed	$3048m,\ 152.4m/s$
θ_0, q_0	Trim Orientation	$0.064rad,\ 0rad/s$
α_0	Trim Angle of Attack	$0.064rad$
$\delta_{t_0}, \delta_{e_0}$	Inputs at Trim	$0.06,\ -0.039rad$
Q	Error Weight Matrix	$diag(1, 49)$
R	Control Weight Matrix	$diag(20, 50)$
$(\lambda_j - \lambda_{j-1})_{max}$	Error Convergence	$1.0e-6$
r	PQP Vector	$[0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$

Fig. 5.1 demonstrates the tracking of h and v respectively for a step setpoint. The controller is able to track both simultaneously with good performance. Tracking performance is slightly better for v. In addition, a rapid change in setpoint between $5s \leq T \leq 15s$ causes slight deterioration in tracking performance, due to the values of N_{p_1}, N_{p_2}.

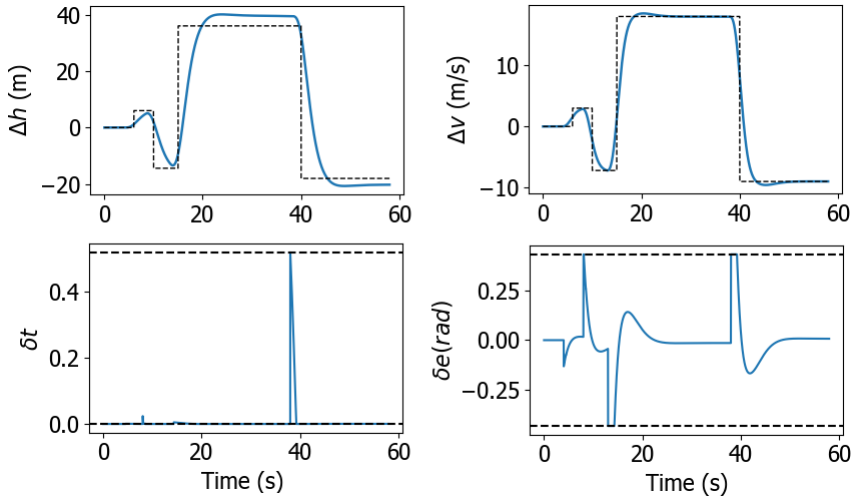

Figure 5.1: Tracking Performance of h and v and absolute actuator positions

For the unconstrained case, the simulation parameters were kept identical. However the R matrix was changed to $diag(2, 50)$, to provide more effort by the elevator. The tracking result for h, v was identical, however the elevator input clearly exceeded its maximum position limit, as demonstrated in Fig. 5.2. In addition, neither of the actuators met their rate limit which is even more crucial. The unconstrained case is an example of utilizing the standard output feedback LQR (using weight matrices Q, R although the cost function is not the exact same). The proposed constrained MPC method is able to match tracking performance whilst taking both actuator position and rate constraints into account. This demonstrates that MPC is the required choice of control method if actuator constraints need to be accounted for in implementation. To elaborate on comparing MPC vs LQR, they are similar in the sense that weighting matrices are defined and an optimization process is used. LQR could satisfy the actuator constraints if the Q, R matrices are tuned properly, however the MPC controller will guarantee this.

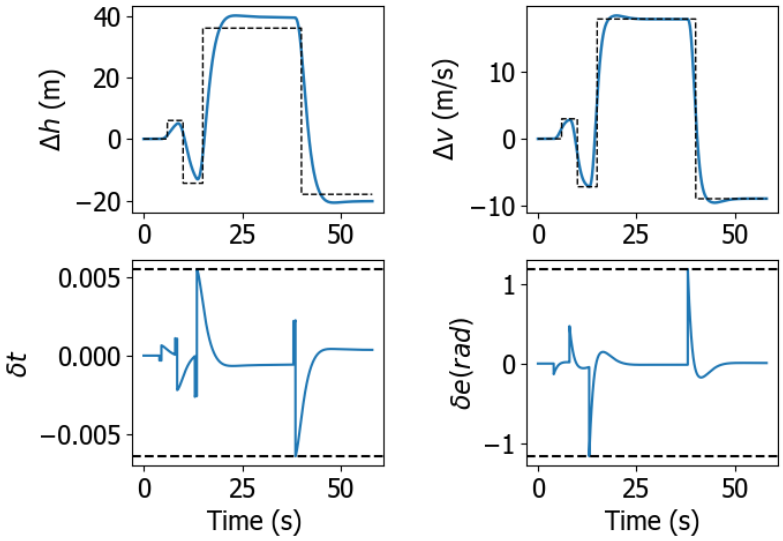

Figure 5.2: Tracking Performance - Unconstrained Case

The incremental changes, i.e. $\Delta(\delta_t), \Delta(\delta_e)$ are displayed in Fig. 5.3. These constraints are also met with the maximum values equalling their upper bounds respectively. Note that absolute values are plotted. Conclusively, the actuator slew rates do not exceed their limits.

The performance of the HQP and PQP solvers is provided in Table 5.3. Clearly, the HQP procedure converges significantly faster, taking an average of only 2 iterations compared to the PQP procedure

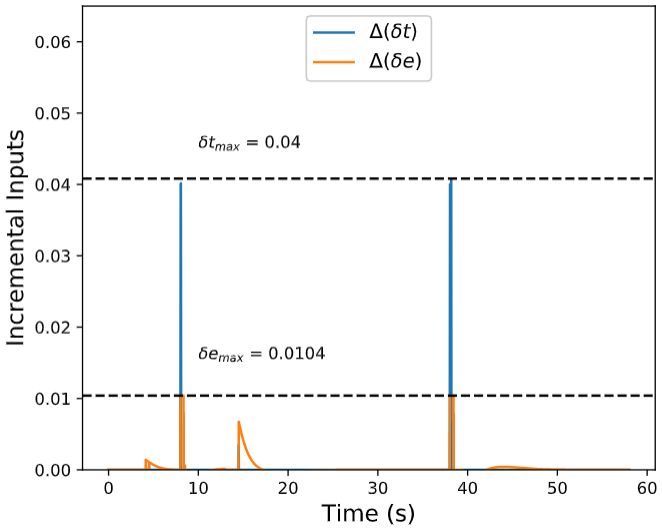

Figure 5.3: Incremental Actuator Movement (Absolute)

which takes roughly seven times longer. However, this simulation was not run on a parallel platform. It could be the case that if the PQP algorithm is run in parallel, solver time can be reduced significantly. Additionally, the PQP algorithm will be faster to implement on hardware as it moves on to the next iteration after obtaining λ, whereas the HQP algorithm which searches for max(λ) at each iteration, requiring the need for an additional maximum search algorithm (See Appendix B). These above factors are critical when choosing which QP solver to run on a flight computer. Note that the number of iterations will drastically increase when there is a setpoint step change.

Table 5.3: QP Solver Performance

Solver Used	**Iterations To Converge**		
	Min	Max	Average
HQP	1.0	177.0	2.05
PQP	6.0	3631	15.55

As mentioned in [10], the prediction horizons N_{p_i} is not set too large or small. A small N_{p_i} may be less than the non-minimum phase time of the system (common for aircraft), posing a risk of destabilizing the closed loop MPC. A large N_{p_i} will intuitively cause a slower response leading to greater steady state error. Additionally, a large N_{pi} will be unable to capture any time varying setpoints over a shorter interval. To verify this behavior, N_{p_i} was set equal for h and v and was

adjusted from $0.5s \rightarrow 1s \rightarrow 2s \rightarrow 3.5s$ respectively. When $N_{p_i} = 0.5s$, oscillations start to occur with significant overshoot (45%). When $N_{p_i} = 1.0s$, the overshoot drops to 6.5%. When $N_{p_i} = 2.0$ or $3.5s$, there is no longer any overshoot due to the increase in the damping ratio, as seen in Fig. 5.4

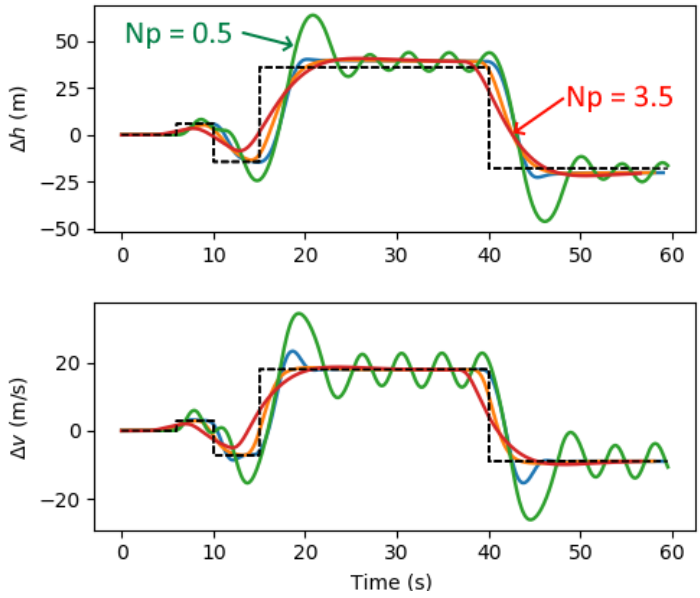

Figure 5.4: Effect of Adjusting the Prediction Horizons

5.1.2 Lateral Model

This is a simulation performed on the F-16 lateral model from 4.2.3. The control variables are ϕ and β (incremental i.e. $\Delta\phi$, $\Delta\beta$). The position and rate constraints on the actuators δ_a, δ_r are presented in Table 5.4. The parameters are presented in Table 5.5. Note that N_{p_1} corresponds to the horizon for ϕ and N_{p_2} corresponds to the horizon for β.

Table 5.4: F-16 Actuator Constraints

Actuator Type	**Constraints**	
	Absolute	***Incremental*** (Δ/s)
Rudder (δ_r)	$-0.523 \leq \delta_r \leq 0.523$	$-2.094 \leq \Delta(\delta_r) \leq 2.094$
Aileron (δ_a)	$-0.3752 \leq \delta_a \leq 0.3752$	$-1.3962 \leq \Delta(\delta_a) \leq 1.3962$

This simulation included a cascaded PID controller for tracking of heading angle ψ. In other words, the setpoint for ψ became the setpoint for ϕ, because the aircraft has to roll to a certain heading!

Table 5.5: Simulation Parameters - Lateral Model

Parameter	***Description***	***Value***
$\Delta T, T$	Sampling, Simulation Time	$0.01s, 60s$
N_{p_1}, N_{p_2}	Prediction Horizons	$2.00s,\ 0.15s$
h_0, v_0	Trim Height & Speed	$3048m,\ 152.4m/s$
θ_0, q_0	Trim Orientation	$0.064rad,\ 0rad/s$
α_0	Trim Angle of Attack	$0.064rad$
$\delta_{a_0}, \delta_{r_0}$	Inputs at Trim	$0.06,\ -0.00rad$
Q	Error Weight Matrix	$diag(150, 10)$
R	Control Weight Matrix	$diag(1, 1)$
$(\lambda_j - \lambda_{j-1})_{max}$	Error Convergence	$1.0e-6$
r	PQP Vector	$[0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$

PID Control is arguably one of the most popular controller methods used in industry. A simple proportional (P) control law for a single CV d is provided by (where d_r is the desired value and d_a is the output for d respectively).

$$U = K_p(d_r - d_a) \tag{114}$$

The disturbances were set to $w = \mathcal{N}(-0.05, 0.002)$ and, $v = \mathcal{N}(0.00,\ 5e-5)$. The aircraft was commanded to track step changes in ψ via the outer PID loop, and the inner MPC loop tracked ϕ whilst regulating $\beta = -0.0523$ (3.0^o). Hence, the PID controller output became the setpoint for ϕ. A precompensator $N = 7.5$ was used for the setpoint ϕ. The maximum permissible value was $\phi = 0.300$ (17.2^o). A block diagram of this controller structure along with the constraints is shown in Fig. 5.5

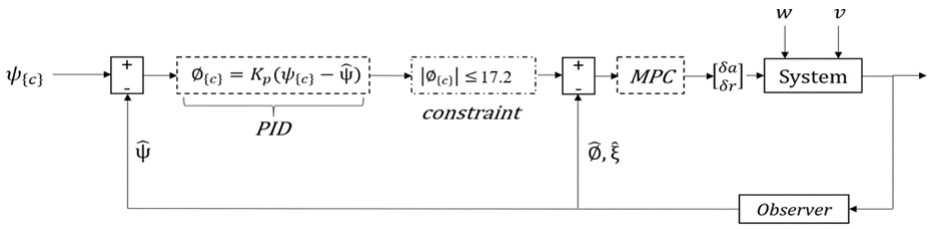

Figure 5.5: Cascaded PID and MPC Control Architecture

The observer parameters K^x, K^w in (43) and (42) were set to:

$$K_x = \begin{bmatrix} 0.4 & 3.5e-07 & -2.1e-15 & 0.00024 & -0.00045 & -3.5e-05 \\ 3.7e-07 & 0.4 & 3.7e-16 & -6.4e-05 & 5.2e-06 & -0.00045 \\ 1.2e-15 & 1.3e-16 & 0.4 & 5.2e-15 & -3e-15 & 2.4e-15 \\ -0.00015 & 2.8e-05 & 1.8e-15 & 0.4 & -0.017 & 0.0052 \\ 0.00072 & -1.1e-05 & -3.1e-15 & 0.011 & 0.4 & -0.0017 \\ 2.8e-05 & 0.00069 & -2.1e-16 & -0.0038 & -0.00071 & 0.4 \\ -8.6e-06 & -5e-07 & 4.1e-16 & 1.3e-05 & -0.00075 & -4.5e-05 \\ 1.1e-06 & -9e-07 & 1.7e-15 & -2.2e-07 & 9.8e-05 & -7.9e-05 \end{bmatrix} \tag{115}$$

$$K_w = \begin{bmatrix} 0.4 & -1.9e-07 & 1.1e-15 & -0.00022 & 0.00051 & 3.4e-05 \\ -2.1e-07 & 0.4 & -2.7e-16 & 5.7e-05 & -6.1e-06 & 0.0005 \\ -2.2e-15 & -3.3e-17 & 0.4 & -3.9e-15 & 1.3e-15 & -2.5e-15 \\ 0.00017 & -3.4e-05 & -4.6e-16 & 0.4 & 0.015 & -0.0048 \\ -0.00066 & 9.8e-06 & 1.4e-15 & -0.012 & 0.4 & 0.0012 \\ -2.9e-05 & -0.00063 & 1.8e-16 & 0.0041 & 0.00019 & 0.4 \end{bmatrix} \tag{116}$$

Fig. 5.6 displays the estimation of $\hat{w}$ from K^w, for two states, β, v. The observer performs well with a very fast settling time of around $0.1s$. Hence, this estimate can now be used to offset the setpoint within the MPC formulation.

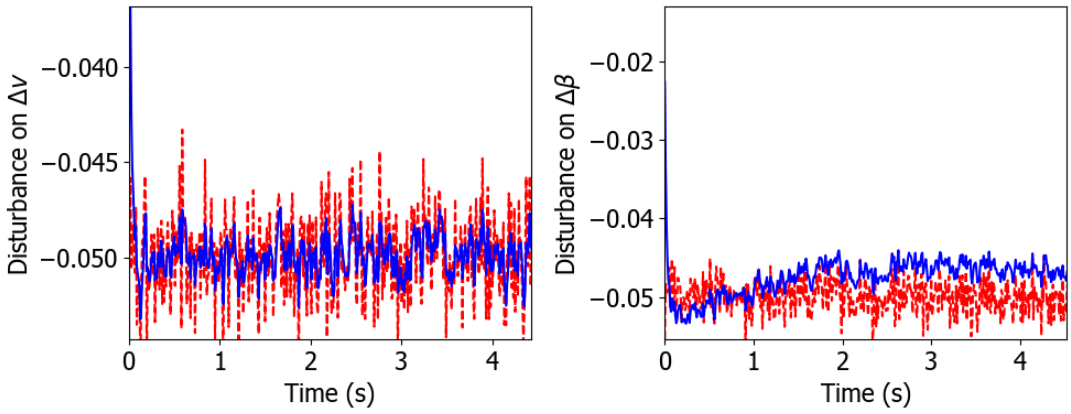

Figure 5.6: Disturbance Observer Performance

Fig. 5.7 displays tracking performance for a step command of $\psi = 30^o$. Perfect tracking is achieved though the settling time is slow due to the constraint imposed on ϕ. The MPC rejects the constant

disturbance and achieve the desired ψ without steady state error. Sideslip angle β is regulated near -3^o. The "Perfect" plot represents the disturbance-free scenario i.e. the situation where $w = \mathcal{N}(0.0, \sigma^2)$.

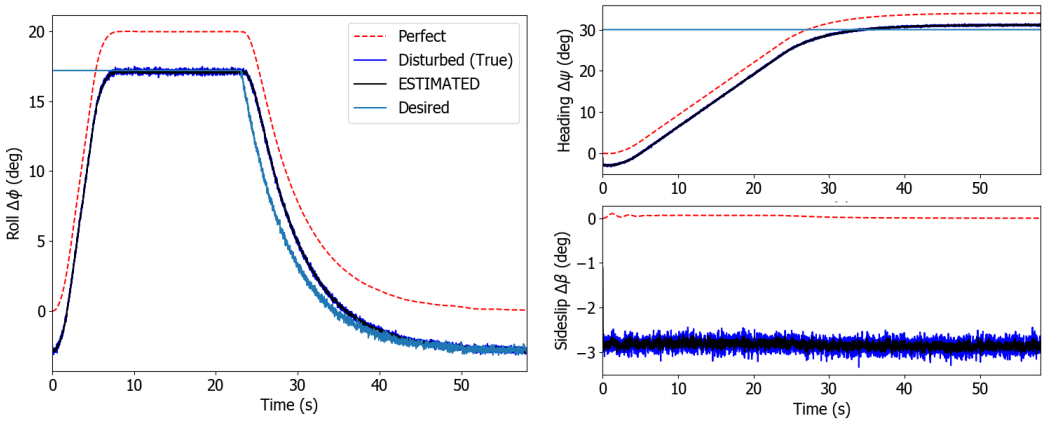

Figure 5.7: Tracking Performance for ψ and ϕ, β

Fig. 5.8 displays the rudder and aileron absolute and incremental positions (in deg). Clearly, both the position and incremental movement constraints are met as both QP solvers ensure this at each time step.

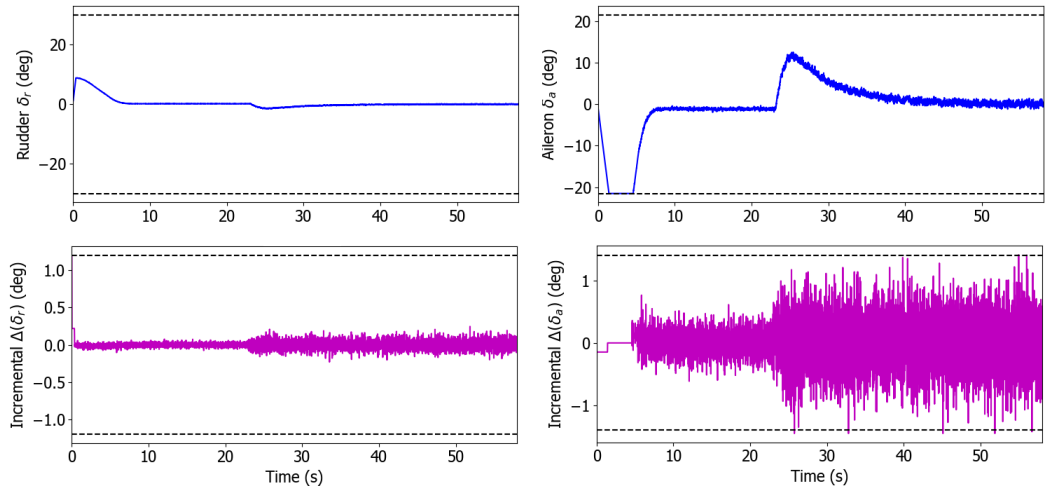

Figure 5.8: Absolute and Incremental Actuator Movement

5.1.3 Combined Longitudinal and Lateral Model

This is a simulation performed on the AFTI F-16 lateral model from 4.2.4. The CVs are $\theta, u, \phi, \beta, r$ (incremental i.e. $\Delta\theta, \Delta u, \Delta\phi, \Delta\beta, \Delta r$). The position and rate constraints on the actuators $\delta_{eR}, \delta_{eL}, \delta_{fR}$, $\delta_{fL}, \delta_c, \delta_r$ are presented in Table 5.6.

Table 5.6: AFTI F-16 Actuator Constraints

Actuator Type	**Constraints**	
	Absolute	***Incremental*** (Δ/s)
Right / Left Horizontal Tail $(\delta_{eR}, \delta_{eL})$	$-0.4363 \leq \delta_{eR}, \delta_{eL} \leq 0.4363$	$-1.0472 \leq \Delta(\delta_{eR}, \delta_{eL}) \leq 1.0472$
Right / Left Flaperon $(\delta_{fR}, \delta_{fL})$	$-0.3752 \leq \delta_{fR}, \delta_{fL} \leq 0.3752$	$-0.9076 \leq \Delta(\delta_{fR}, \delta_{fL}) \leq 0.9076$
Canards (δ_c)	$-0.3752 \leq \delta_c \leq 0.3752$	$-1.8849 \leq \Delta(\delta_c) \leq 1.8849$
Rudder (δ_r)	$-0.3752 \leq \delta_r \leq 0.3752$	$-2.0944 \leq \Delta(\delta_r) \leq 2.0944$

When the simulation was attempted using these constraints, the QP became infeasible and a valid solution could not be found for the constrained input U. As a result, the incremental constraints for only the right and left horizontal tail had to be softened by a factor of 20 from $-1.0472/s \leq \Delta(\delta_{eR}, \delta_{eL}) \leq 1.0472/s \longrightarrow -20.944/s \leq \Delta(\delta_{eR}, \delta_{eL}) \leq 20.944/s$. The remaining constraints were left as is and now the QP remained feasible throughout the simulation. With the MPC parameters in Table 5.7, the closed loop poles are stabilized, as demonstrated in Fig. 5.9.

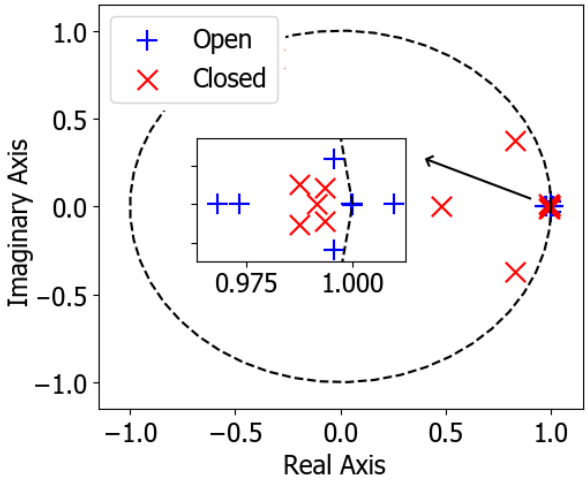

Figure 5.9: AFTI F-16: Open Loop vs. Closed Loop

In regards to the setpoints for the 5 CVs $(\theta, u, \phi, \beta, r)$ some were tasked to regulate around 0 while others were set for tracking. To begin, the pitch angle θ and sideslip β were set for tracking to a

small value ($0.04rad$,$0.10rad$) at $30s$ intervals. The yaw rate r and speed u were regulated to 0, i.e. no change from the linearized operating point. Lastly, the roll angle ϕ was commanded to $0.25rad$ at $30s$ intervals. The tracking / regulation performance for the CVs θ, β, u, r is shown in Fig. 5.10 and that for the CV ϕ is in Fig. 5.11. The performance is good for θ, β with little to no overshoot.

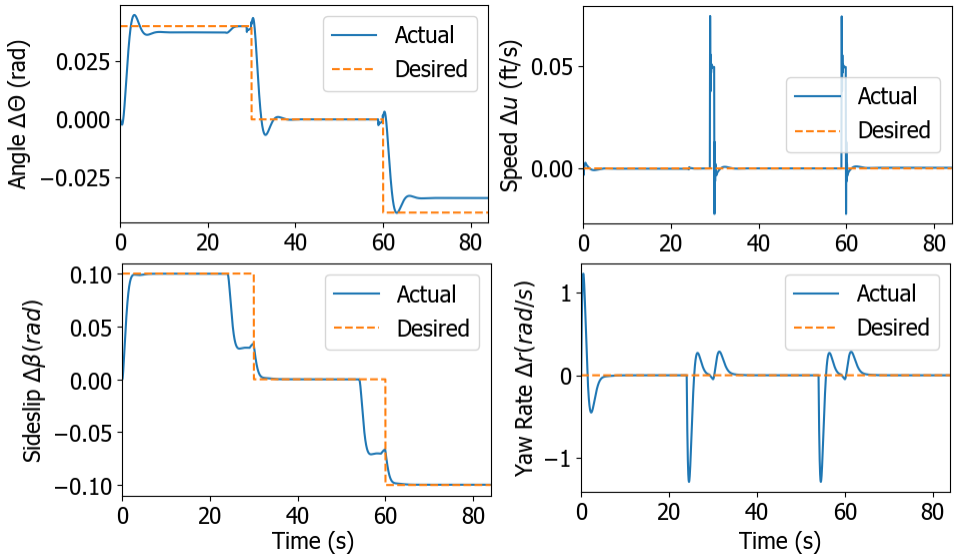

Figure 5.10: Tracking Performance - AFTI F-16

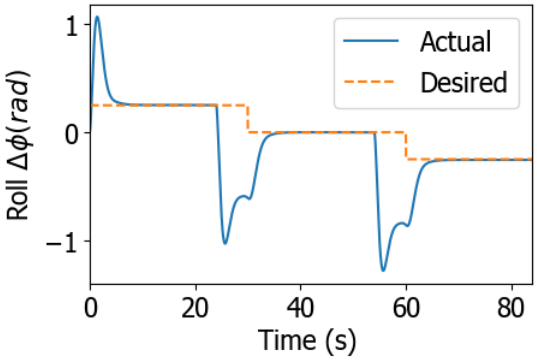

Figure 5.11: Tracking Performance - AFTI F-16

Since there is a setpoint change at $30s$ and $60s$ respectively, this will create a sharp change in u and r. For u the change is very minimal however there is a large overshoot for r. Finally, for ϕ, steady state tracking is good however there is a large overshoot. By observing the AFTI model from 4.2.4, the longitudinal and lateral states are decoupled in A_c, however there is coupling between the control in B_c. This creates the overshoot in some of the CVs. To reduce this overshoot, the values

for N_{pi} can be increased, however this would then reduce the settling time.

Table 5.7: Simulation Parameters - AFTI F-16 Model

Parameter	***Description***	***Value***
$\Delta T, T$	Sampling, Simulation Time	$0.01s, 90s$
$N_{p_1}, N_{p_2}, N_{p_3}, N_{p_4}, N_{p_5}$	Prediction Horizons	$0.15s,\ 1.15s,\ 6.00s,\ 0.10s,\ 1.00s$
h_0, v_0	Trim Height & Speed	$6096m,\ 308.4m/s$
θ_0, q_0	Trim Orientation	$0.00rad,\ 0rad/s$
α_0	Trim Angle of Attack	$0.032rad$
$\delta_{eR0}, \delta_{eL_0}, \delta_{fR0}, \delta_{fL0}$	Inputs at Trim	$0, 0, 0, 0rad$
δ_{c0}, δ_{r0}	Inputs at Trim	$0, 0rad$
Q	Error Weight Matrix	$diag(50, 50, 50, 50, 50)$
R	Control Weight Matrix	$diag(1, 1, 1, 1, 1, 1)$
$(\lambda_j - \lambda_{j-1})_{max}$	Error Convergence	$1.0e-6$
r	PQP Vector	$[0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$

Figs 5.12 and 5.13 display the inputs and incremental inputs. The constrained QP once again performs successfully with all position and rate constraints satisfied.

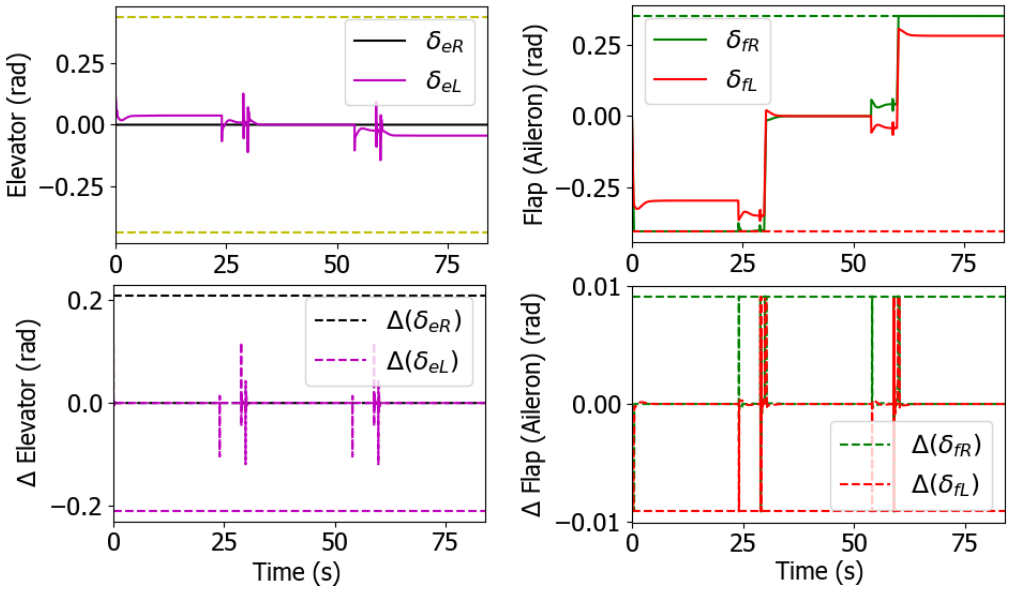

Figure 5.12: Inputs and Incremental Inputs - AFTI F-16

To conclude this section, the AFTI F-16 simulation demonstrated that generally when designing linear controllers, it is not the best idea to couple the longitudinal and lateral control inputs as this can cause suboptimal performace, leading to issues such as overshoot and slow settling time for specific control variables, depending upon how the controller is configured. Using seperate

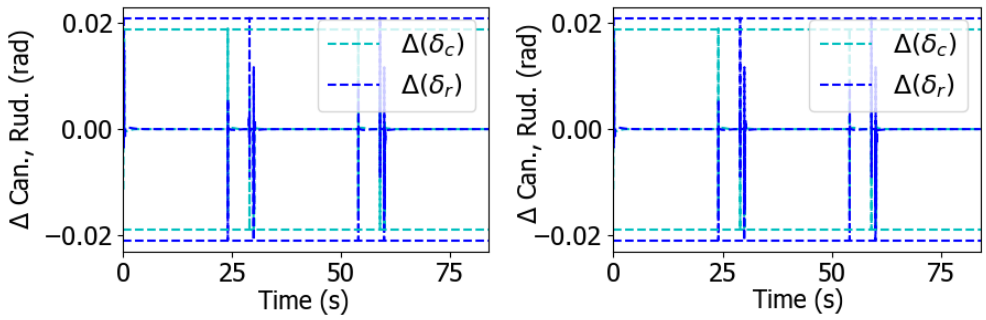

Figure 5.13: Inputs and Incremental Inputs - AFTI F-16

longitudinal and lateral MPC based controllers provide the best results.

5.2 Fault Conditions

This section contains simulations for both the integral action and the efficient MPC method, this time in the presence of faults. The purpose of these simulations was to demonstrate the performance of the fault observer along with the reconfiguration method for the efficient MPC algorithm. The model used was the generic longitudinal model from 4.2.5. These simulations were also performed with Python, and successful performance of the FDD algorithm is achieved. Note that the results are shown side by side for the efficient and integral action MPC methods because the same setpoints and CVs were used, for easier comparison.

5.2.1 Observer Design and MPC Parameters

The parameters for the fault diagnosis observer discussed in 2.6 were set as follows:

$$L = \begin{bmatrix} 0.63 & -0.0062 & -0.023 & -0.27 & 0.078 \\ -0.012 & 1.00 & 1.80 & 0.0068 & 0.0012 \\ -0.00045 & 0.064 & 0.45 & 0.00092 & 4.2e-5 \\ -0.011 & 0.0083 & 0.026 & 0.53 & -0.087 \\ 0.078 & 0.025 & -0.0029 & -2.1 & 1.20 \end{bmatrix}$$

$$D = \begin{bmatrix} 0.2755 & -0.0009 & -0.0007 & 0.014 & -0.0031 \\ 0.0279 & -2.3998 & -3.103 & -0.0098 & -0.014 \end{bmatrix}$$

$$\epsilon_1 = 1.995,\ \epsilon_2 = 0.018 \leftarrow \text{Integral-Action MPC}$$

$$\epsilon_1 = 15.995,\ \epsilon_2 = 0.400 \leftarrow \text{Efficient MPC}$$

In both MPC simulations, the following faults were injected at the same times, and are listed in Table 5.12. The LOE for both actuators was changed after a random time interval to mimic faults which could potentially occur in real life.

Table 5.8: Actuator Faults

Time (s)	**LOE:** (δ_t)	**LOE:** (δ_e)
1.60	-0.20	-0.45
8.25	-0.50	-0.75
12.25-end	-0.65	-0.55

The position and slew (incremental) constraints of both inputs δ_t, δ_e are provided in Table 5.9.

Table 5.9: Actuator Constraints

Actuator Type	**Constraints**	
	Absolute	**Incremental** (Δ/s)
Thrust (δ_t^{a})	$-0.2 \leq \delta_t \leq 0.8$	$-0.8 \leq \Delta(\delta_t) \leq 0.8$
Elevator (δ_e)	$-0.4363 \leq \delta_e \leq 0.4363$	$-0.52 \leq \Delta(\delta_e) \leq 0.52$

[a]Approximate Value

Table 5.10 provides the simulation parameters used for both model predictive control schemes. To ensure the most accurate results, parameters such as the discretization time dT and the weighting matrices Q, R along with the prediction and control horizons N_p, N_c were adjusted for each method. The moving window filters M_v, M_{v2} have to be appropriately scaled up for the efficient MPC since dT is scaled down.

5.2.2 Longitudinal Model - Integral Action vs Efficient MPC

The model used was taken from 4.2.5. The aircraft was initially commanded to track a step change in vertical velocity of $\Delta u_c = 25m/s$ and after $9.0s$, $\Delta u_c = 20m/s$. It was also commanded to track an increasing time varying setpoint for the height Δh, which decreased after $9.0s$. Fig. 5.14

Table 5.10: Integral Action and Efficient MPC Parameters

Parameter	Integral Action MPC	Efficient MPC
dT(s)	0.05	0.01
Time (s)	20.0	45.0
$\Delta(\delta_t)$	0.04	0.008
$\Delta(\delta_e)$	0.026	0.0052
N_p, N_c	30.0, 2.0 (1.5s, 0.1s)	3.0, 1.0 (3.0s, 0.01s)
Q	$diag(1, 1)$	$diag(10, 10)$
R	$diag(1, 1)$	$diag(4, 4)$
M_v, M_{v2}	$k = 8, k = 32$	$k = 40, k = 160$

is the nominal closed loop discrete time pole plot for both MPC methods. As required, the system is stable with the magnitude of all poles less than 1.

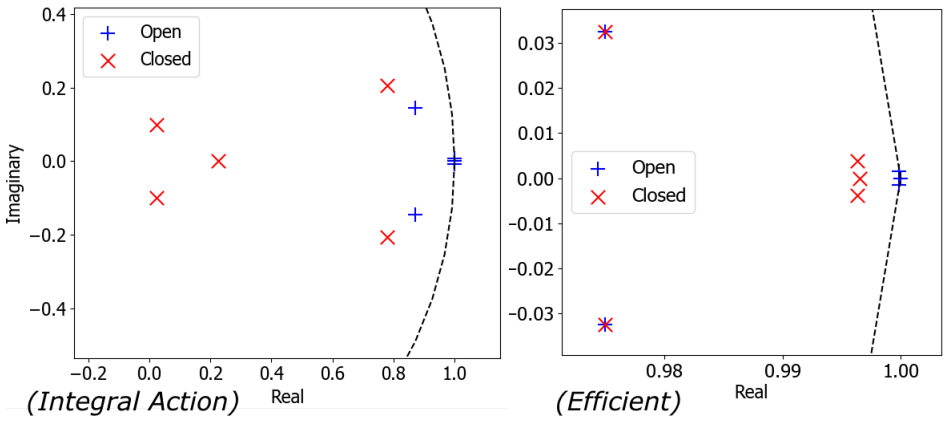

Figure 5.14: Pole Zero Plot - Efficient vs Integral Action Closed Loop MPC

Figure 5.15 displays the tracking performance for the CVs height h and the forward velocity u for both MPC methods. Good tracking is achieved. In addition, the configured state observer displays good performance yielding perfect estimates for the CVs, and correspondingly for the other states as well.

Figs. 5.16 and 5.17 shows the control inputs from δ_t and δ_e along with the incremental inputs $\Delta(\delta_t)$ and $\Delta(\delta_e)$. All inputs meet their constraints, thus demonstrating the successful performance of the QP solver for both simulations.

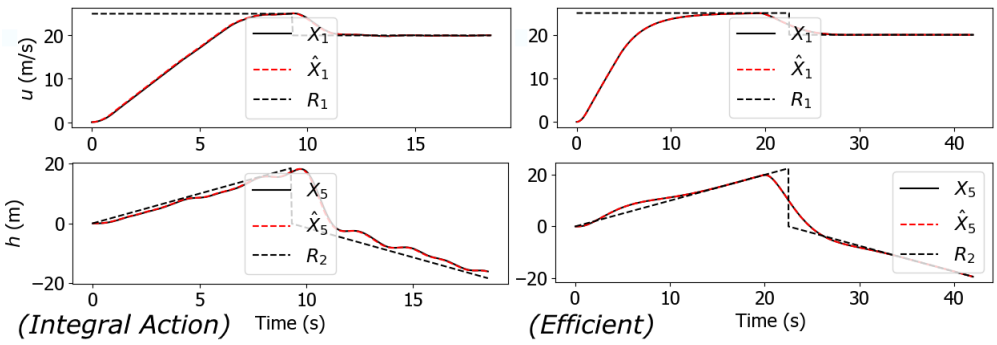

Figure 5.15: Tracking Performance - Efficient vs Integral Action Closed Loop MPC

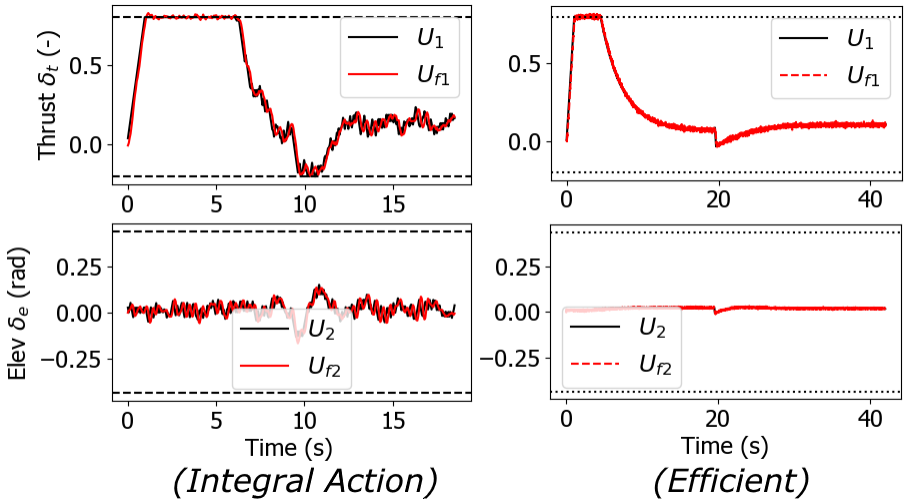

Figure 5.16: Inputs - Efficient vs Integral Action Closed Loop MPC

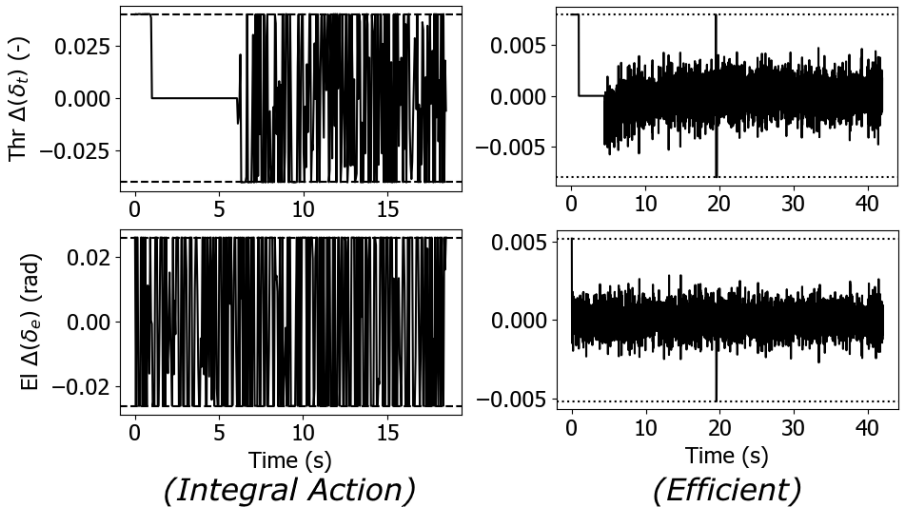

Figure 5.17: Incremental Inputs - Efficient vs Integral Action Closed Loop MPC

Fig. 5.18 provides the tracking performance for both MPC methods in the presence of faults, without any reconfiguration. As displayed, the integral action MPC scheme does a very good job of stabilizing the system once faults occur and ensuring that the CVs track the setpoints. This is because the optimization variable is ΔU rather than U, so in the event of faults, the states $\hat{X}$ do not deviate a large amount from their nominal values. Furthermore, this simulation reinforces the fact that this MPC algorithm has implicit fault tolerant capability, as demonstrated in [11]. As long as the faults are not severe and the resulting closed loop system is stable, the above holds true.

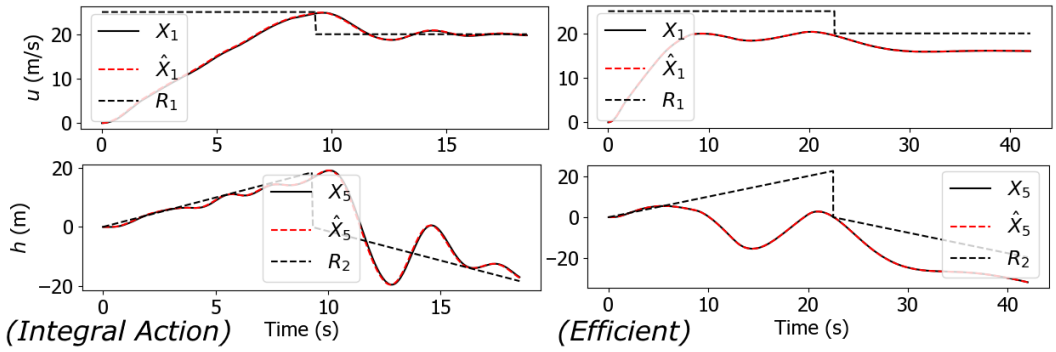

Figure 5.18: Tracking in Fault Conditions - Efficient vs Integral Action MPC

Looking at the efficient MPC results, the states deviate a large amount from their fault-free values in Fig. 5.19. The values of $\hat{X}$ go beyond the domain of the MPC optimization, and as a result u and h never reach their setpoint value. This clearly indicates that the efficient MPC has a significantly lesser degree of implicit fault tolerant capability and there should be a reconfiguration mechanism present. Fig. 5.19 displays the results when the controller is reconfigured using 17. Both u, h reach their setpoints without much delay. Note that after $12.0s$, the LOE faults are still present at 65% for δ_t and 55% for δ_e respectively.

Fig. 5.20 shows the performance of the moving average filter algorithm i.e. the filtered inputs for both simulations. The moving average filter removes a lot of noise from the input signal, increasing the accuracy of the calculation of $\hat{f}$. Lastly, Fig. 5.21 is the estimation of $\hat{f}$. The estimation is accurate with a small delay (due to the value of M_{v2}). This indicates that the value for M_{v2} should not be too small as it will affect the accuracy, albeit not too large to increase the delay of the reported value of $\hat{f}$.

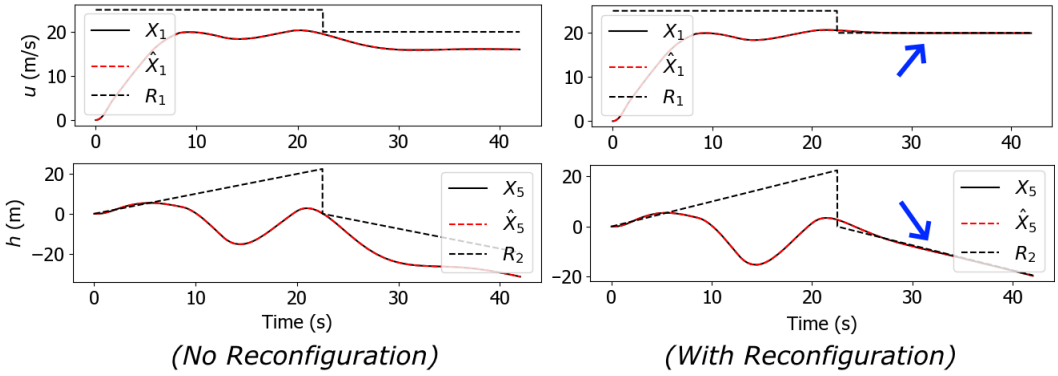

Figure 5.19: Reconfiguration - Efficient MPC

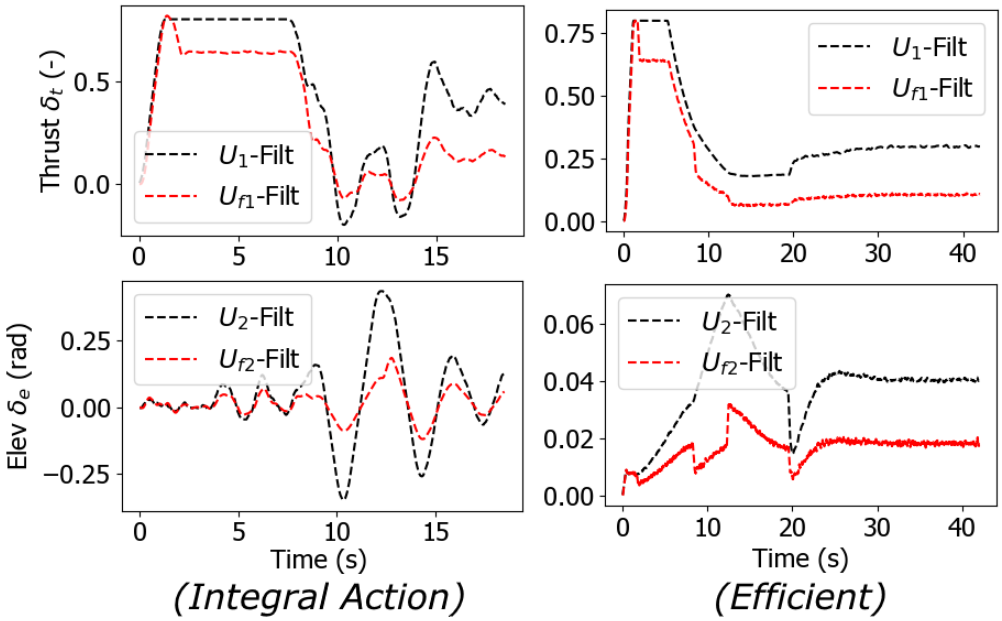

Figure 5.20: Moving Average Filter Performance

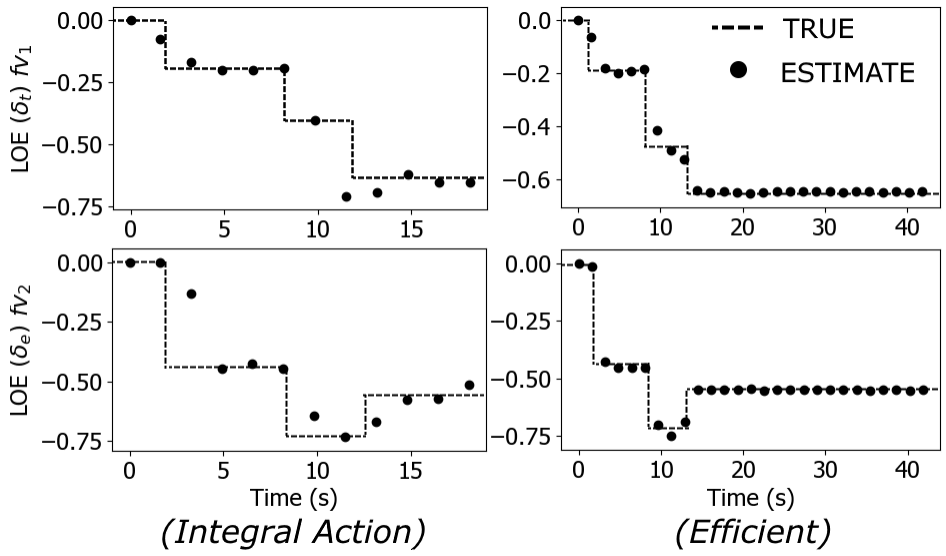

Figure 5.21: Fault Estimation - True vs. Estimated faults

5.2.3 Efficient vs. Integral Action MPC - Detailed Comparison

Looking at the simulation results from 5.2.2, Figures 5.15 and 5.16 provide a great comparison of how both the MPC methods perform. Firstly, the simulation run time was about $180s$ for the efficient MPC simulation and $1175s$ for the integral action MPC. This is inherently due to the larger dimension for the latter method and it's quadratic optimizer. In Fig. 5.15, the settling time for Δu is similar at $\simeq 8.0s$, along with very similar control inputs from δ_t, δ_e. This is because the prediction horizons were carefully tuned to yield similar performance.

For the fault conditions in 5.18, the integral action MPC has a much better degree of stability compared to the efficient MPC (without reconfiguration). The maximum observed tracking error for the eMPC is as large as 210% at (t = $14s$), as for the iMPC, it does not exceed 35%. Furthermore towards the end of the simulation ($t = 25s+$), the steady state error for the integral action MPC converges towards zero, with minimal damping, while for the efficient MPC the controller has to be reconfigured to reach zero.

In regards to the observer performance for both methods (Fig. 5.21), due to the tuning of the ϵ_i parameter, the resulting accuracy is near identical for both methods. Note that the moving FFT ratio window length (M_v) is identical for both methods, because the sampling time is $0.05s, 0.01s$ respectively i.e ($k = 32, k = 160$) in Table 5.10. To conclude this simulation demonstrates that the

efficient MPC can yield similar results to the integral action MPC if it is tuned properly and reconfigured in the presence of faults. However the reconfiguration does have limits as U is bounded, so the integral action could potentially handle faults of a larger magnitude.

5.3 Closed Loop Guidance Design

5.3.1 Reduced Order Model - Integral Action vs. Efficient MPC

For this final simulation, the model used was the reduced order longitudinal and lateral dynamics from 4.2.6. A closed loop guidance model was also developed where the aircraft was commanded to follow a predefined trajectory. Both MPC methods were utilized. To begin, the CVs (control loop setpoints) for the longitudinal and lateral model were $Z_{lon} = [V_T \ \theta]^T$ and $Z_{lat} = [\beta \ \phi]^T$ respectively. Table 5.11 displays the actuator constraints for this model.

Table 5.11: Actuator Constraints - Reduced Order Model

Actuator Type	**Constraints**	
	Absolute	**Incremental** (Δ/s)
Thrust (δ_t^a)	$-0.2 \leq \delta_t \leq 0.8$	$-0.8 \leq \Delta(\delta_t) \leq 0.8$
Elevator (δ_e)	$-0.4363 \leq \delta_e \leq 0.4363$	$-1.04 \leq \Delta(\delta_e) \leq 1.04$
Aileron (δ_a)	$-0.3752 \leq \delta_t \leq 0.3752$	$-1.38 \leq \Delta(\delta_t) \leq 1.38$
Rudder (δ_r)	$-0.4363 \leq \delta_e \leq 0.4363$	$-2.08 \leq \Delta(\delta_e) \leq 2.08$

[a]Approximate Value

Two separate simulations were performed. Section 5.3.2 describes the performance for tracking a generic time varying setpoints in fault-free and fault conditions, and Section 5.3.3 describes results where the aircraft is commanded to follow a trajectory. Parameters are shown in Table 5.12.

The parameters for the fault diagnosis observer discussed in 2.6 were set as follows:

$$L_{lon} = \begin{bmatrix} 0.91 & -0.11 & -1.0 & -0.065 \\ -0.01 & 0.58 & -0.0047 & 0.011 \\ -0.095 & -0.009 & 0.57 & 0.033 \\ -0.00045 & -0.14 & 0.0046 & 0.57 \end{bmatrix}$$

Table 5.12: Integral Action and Efficient MPC Parameters

Parameter	**I-MPC**		**E-MPC**	
	Simulation 1	*Simulation 2*	*Simulation 1*	*Simulation 2*
dT (s)	0.05		0.01	
Time (s)	80.0	$50.0^a, 190.0^b$	80.0	$54.0^a, 195.0^b$
$\Delta(\delta_t), \Delta(\delta_e)$	$0.04,\ 0.052$		$0.008,\ 0.0104$	
$\Delta(\delta_a), \Delta(\delta_r)$	$0.069,\ 0.104$		$0.0138,\ 0.0208$	
N_{plon}, N_{clon} (s)	$0.45,\ 0.05$		$3.50,\ 0.01$	
N_{plat}, N_{clat} (s)	$10.50,\ 0.00$		$2.20,\ 0.01$	
Q_{lon}, R_{lon}	$diag(100, 900),\ diag(1, 1)$		$diag(150, 150),\ diag(1, 0.1)$	
Q_{lat}, R_{lat}	$diag(150, 150),\ diag(1, 1)$		$diag(150, 150),\ diag(1, 0.1)$	
M_v, M_{v2}	$k = 8, k = 32$		$k = 40, k = 160$	

[a]Straight Line Trajectory, [b]Helical Trajectory

$$D_{lon} = \begin{bmatrix} 0.9857 & 0.0005 & 0.0148 & 0.0015 \\ 0.0052 & -0.0002 & -0.0004 & -0.008 \end{bmatrix}$$

$$L_{lat} = \begin{bmatrix} 0.87 & 0.0027 & -0.0073 & -0.041 \\ -0.027 & 0.89 & 0.043 & 0.0037 \\ -1.1 & 9.2e-4 & 0.8 & 0.045 \\ 0.34 & 4.4e-4 & -0.0084 & 0.87 \end{bmatrix}$$

$$D_{lat} = \begin{bmatrix} 0.002 & -0.0007 & -0.0254 & -0.0012 \\ 1e-4 & 1e-4 & 0.0043 & -0.0024 \end{bmatrix}$$

$\epsilon_1 = 0.075,\ \epsilon_2 = 350.0 \leftarrow$ Integral-Action MPC (Longitudinal)

$\epsilon_1 = 150.0,\ \epsilon_2 = 1200.0 \leftarrow$ Integral-Action MPC (Lateral)

$\epsilon_1 = 0.50,\ \epsilon_2 = 750.0 \leftarrow$ Efficient MPC (Longitudinal)

$\epsilon_1 = 250.0,\ \epsilon_2 = 2050.0 \leftarrow$ Efficient MPC (Lateral)

The discrete time (z-plane) pole zero plot of the closed loop stable lateral and longitudinal model for both methods is provided in Fig. 5.22. The longitudinal model is at the top and lateral is at the bottom. Clearly, the closed loop system is stabilized with the selected prediction and control horizons, along with the weight matrices in Table 5.12. In particular, the unstable or marginally stable poles are brought back within the unit circle.

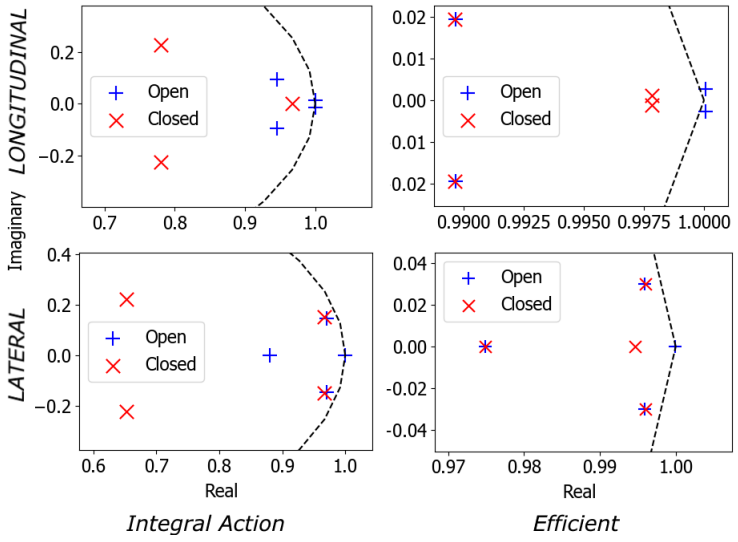

Figure 5.22: Closed Loop Longitudinal and Lateral PZ Plot

5.3.2 Simulation 1 - MPC Performance

The objective of Simulation 1 was to determine the performance of the controller whilst tracking a setpoint varying with time. Usually in autonomous flight control applications, the desired pitch and roll (θ, ϕ) change with time and are calculated based on a reference trajectory. Tracking of Z_{lon} in fault free conditions is shown in Fig. 5.23.

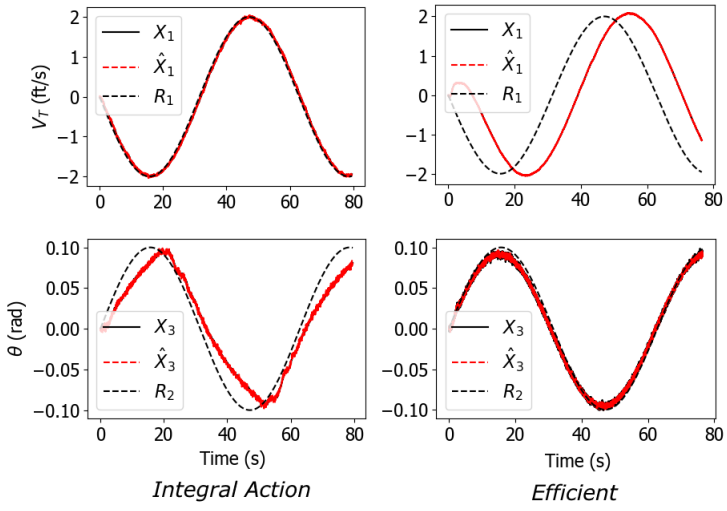

Figure 5.23: Longitudinal Controller Performance - Fault Free

The setpoint for V_T is changed as a function of θ. Note that these are incremental states, i.e. the actual flight speed only varies between $150 - 154m/s$ (the operating point is at $152m/s$). Tracking

for θ is good, with a small inherent time delay due to the constrained inputs and the prediction horizon. Next, the tracking of Z_{lat} in fault free conditions is shown in Fig. 5.24. The key objective is to track ϕ as closely as possible whilst regulating $\beta = 0$, to minimize shear forces on the aircraft. As the controller does a good job of tracking the roll angle, this will lead to a similar tracking performance for the heading angle ψ, demonstrated in 5.3.3.

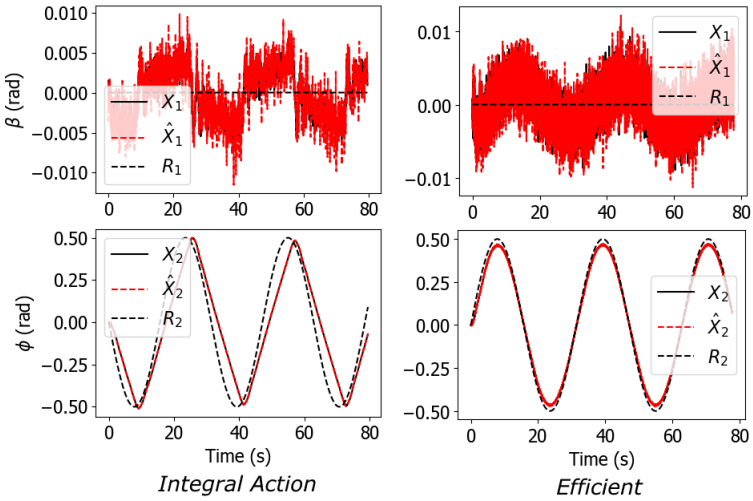

Figure 5.24: Lateral Controller Performance - Fault Free

In the interest of time, the inputs and incremental inputs are not shown for the fault free conditions as all inputs met their operational constraints during all simulations. Next, the tracking for Z_{lon} and Z_{lat} was re-evaluated whilst in the presence of faults. As explained in 3.3, the reconfiguration is only done for the efficient MPC algorithm as the integral action MPC has a degree of implicit fault tolerant capability. For U_{lon}, a 50% LOE was set for δ_t at the one-third point in the simulation (efficient MPC) and a 50% LOE was set for δ_e at the halfway point in the simulation (integral action MPC) . For U_{lat}, a 50% LOE was set for δ_a at the simulation halfway point. In all situations, the closed loop MPC remains stable. Figs 5.25 and 5.26 show tracking performance for the longitudinal and lateral controller respectively, in the presence of faults.

In Fig. 5.25, for the integral action MPC, since there is an LOE for δ_e, this has a big impact on the tracking for θ as intuitively, the elevator directly controls the pitch angle of the aircraft. Although the controller is compromised, it is still stable with θ remaining between its constraint of $-0.1 < \theta < 0.1$. As demonstrated by the simulation for the efficient MPC, the LOE for δ_t has

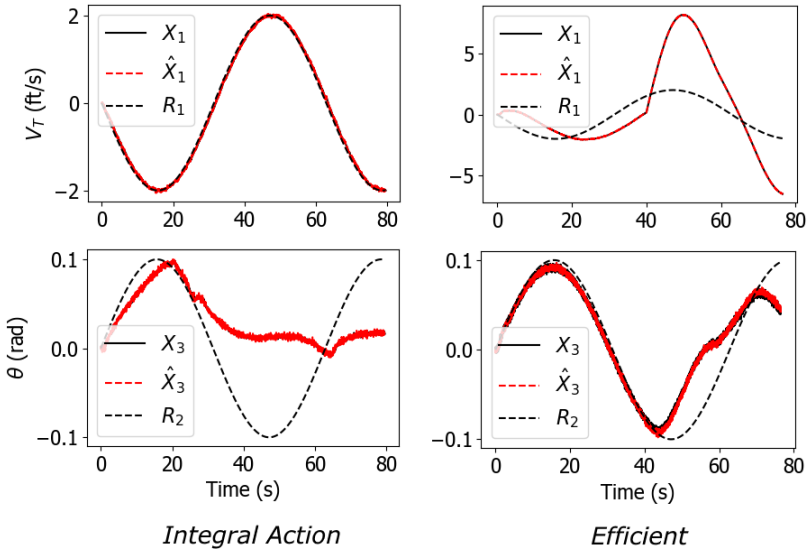

Figure 5.25: Longitudinal Controller Performance - Faults

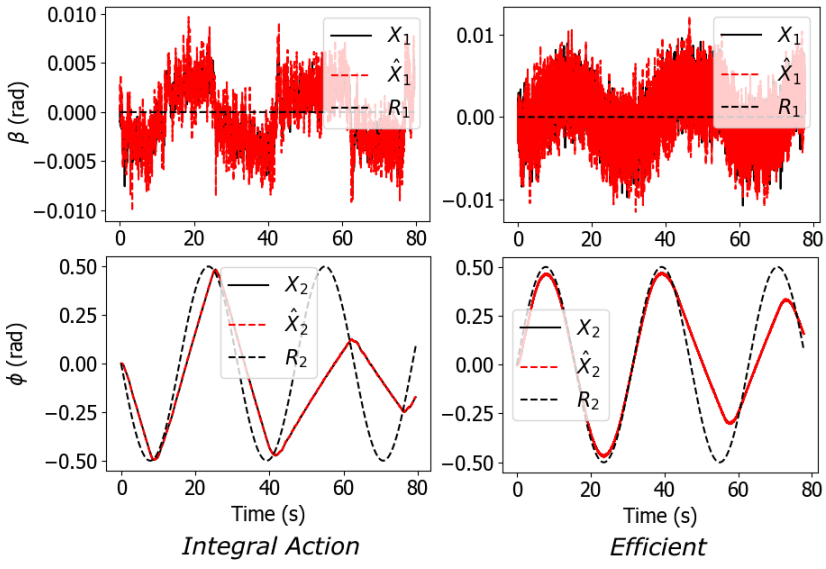

Figure 5.26: Lateral Controller Performance - Faults

a large impact on V_T. However unlike the previous situation, it does also have a little impact on θ. In both cases, the CVs are still maintained within a specific range due to the reconfiguration mechanism. In Fig. 5.26, as the aileron has the biggest impact on the roll angle ϕ, the resulting performance is compromised. The performance for β is unaffected. However, the efficent MPC does a better job of attempting to minimize the error as much as possible due to its reconfiguration mechanism. Once again, the controller in the presence of faults is stable in both cases. Lastly, the inputs and incremental inputs in the presence of faults are shown in Figs. 5.27 and 5.28 to demonstrate that the observer works perfectly with correct estimation of $\hat{U}_{f-lon}$ and $\hat{U}_{f-lat}$.

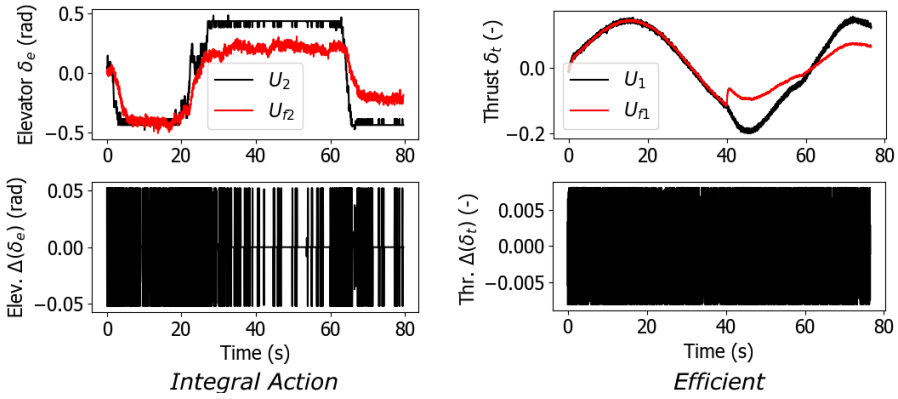

Figure 5.27: Longitudinal Inputs with Estimation

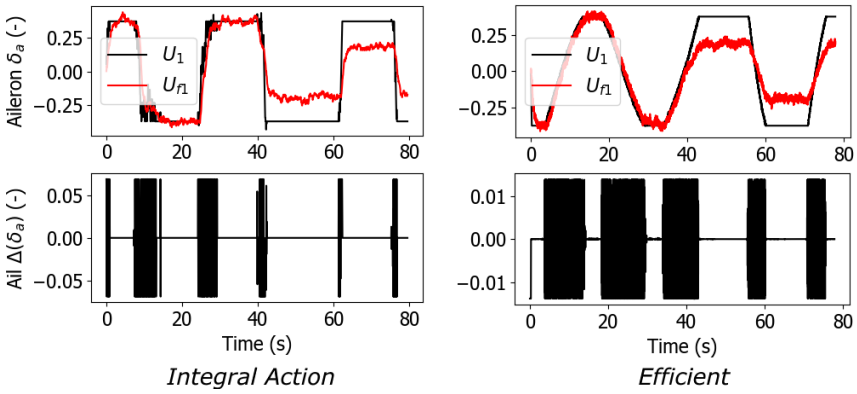

Figure 5.28: Lateral Inputs with Estimation

5.3.3 Simulation 2 - Closed Loop Guidance

The final simulation consisted of a complete closed loop guidance design using these decoupled dynamics. A reference trajectory was pre-generated and then a closed loop guidance method was used to generate the required attitude commands. The MPC controllers were part of the innermost loop which converted the setpoints to the required actuator inputs. A block diagram is provided.

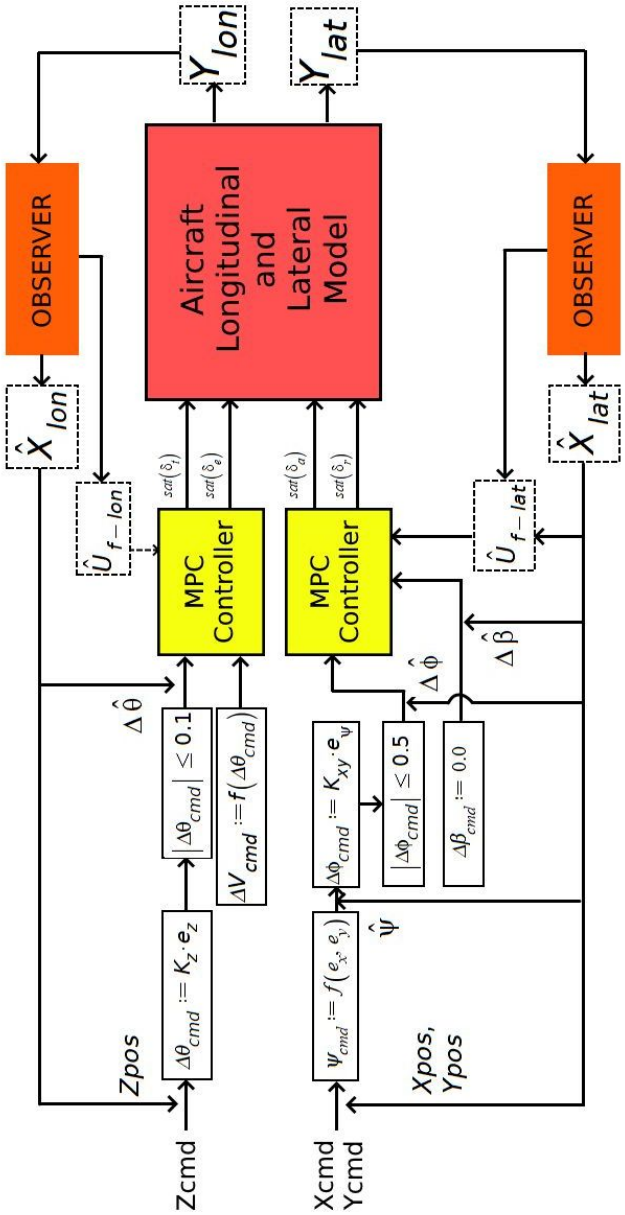

Figure 5.29: Guidance and Control Design Block Diagram

The algorithm for the outer loop guidance simulation to calculate the setpoint commands for ϕ, V_T, ψ, β (combined longitudinal and lateral) is as follows (Algorithm 2). The outermost loops are two PID loops which convert the error between the positions to the required roll, pitch and heading angles. Hence this is a form of attitude control based on position, which is widely used in all aerospace systems. A simple proportional (P) controller works well for the outer loop.

Algorithm 2 Closed Loop Guidance

Require: $X_{cmd}, Y_{cmd}, Z_{cmd}$ ▷ All waypoint coordinates
OP ▷ Operating point vector (both lateral / longitudinal
Ensure: $\Delta T > 0, T > 0$
$k = 0$
$k_f = T/\Delta T$
for $k = 0 \rightarrow k_f$ **do** ▷ k is the time step
$X_{lon,k+1} = AX_{lon,k} + BU_{lon,k}$ ▷ Longitudinal Simulation
$Y_{lon,k+1} = CX_{lon,k}$
$X_{lat,k+1} = AX_{lat,k} + BU_{lat,k}$ ▷ Lateral Simulation
$Y_{lat,k+1} = CX_{lat,k}$
Estimate $\hat{X}_{lon}$, $\hat{X}_{lat}$ from observer
Get positions $X_{true}, Z_{true}, Z_{true}$ from OP, X_{lon}, X_{lat} ▷ Use equations of motion
$e_z = Z_{cmd} - Z_{true}$
$\Delta\theta_{cmd} = K_z \cdot f_\theta(e_z)$ ▷ Controller for $\Delta\theta$
$e_x = X_{cmd} - X_{true}$
$e_y = Y_{cmd} - Y_{true}$
$\psi_{cmd} = atan2(e_y, e_x)$ ▷ Function to get heading
$e_\psi = \psi_{cmd} - \hat{\psi}$, $e_y = Y_{cmd} - Y_{true}$
$\phi_{cmd} = K_{xy} \cdot e_\psi$ ▷ PID Controller for $\Delta\psi$
$|\Delta\theta| \leq 0.1rad$ and $|\Delta\phi| \leq 0.5rad$
$\beta_{cmd} = 0$
$\Delta V_{cmd} = -20 \cdot \Delta\theta_{cmd}$
Send $\Delta\theta_{cmd}, \Delta\phi_{cmd}, \beta_{cmd}, \Delta V_{cmd}$ to MPC controller
Obtain required $U_{lat,(k+1)} = [\delta_a \ \delta_r]^T$
Obtain required $U_{lon,(k+1)} = [\delta_T \ \delta_e]^T$
$k = k + 1$
end for

The P gain values were set to $K_z = \pm -1$ depending on the sign of e_z and $K_{xy} = 1.15$. Furthermore, the required pitch attitude function f_θ is:

$$f_\theta = acos\left(\frac{\sqrt{e_x^2 + e_y^2}}{\sqrt{e_x^2 + e_y^2 + e_z^2}}\right) \tag{117}$$

Two reference trajectories were defined, the first being a single waypoint defined at an arbitrary XYZ position. The second trajectory is a discretized semi helical trajectory with the aircraft commanded to climb and then descend at the halfway point. Performance was evaluated in both fault free and fault conditions. Once again in the interest of time, the actuator inputs will only be shown for the fault conditions in the single waypoint simulation. The trajectory tracking performance for the single waypoint in both fault free and fault conditions is shown in Figs. 5.30 and 5.31. In regards to the faults, a 50% LOE was applied to δ_e, δ_a at the halfway point.

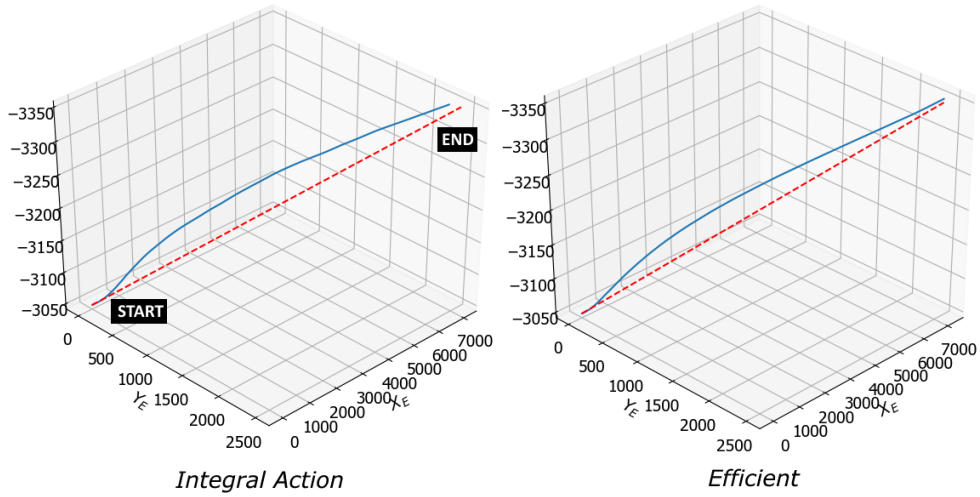

Figure 5.30: Single Waypoint - Fault Free Tracking

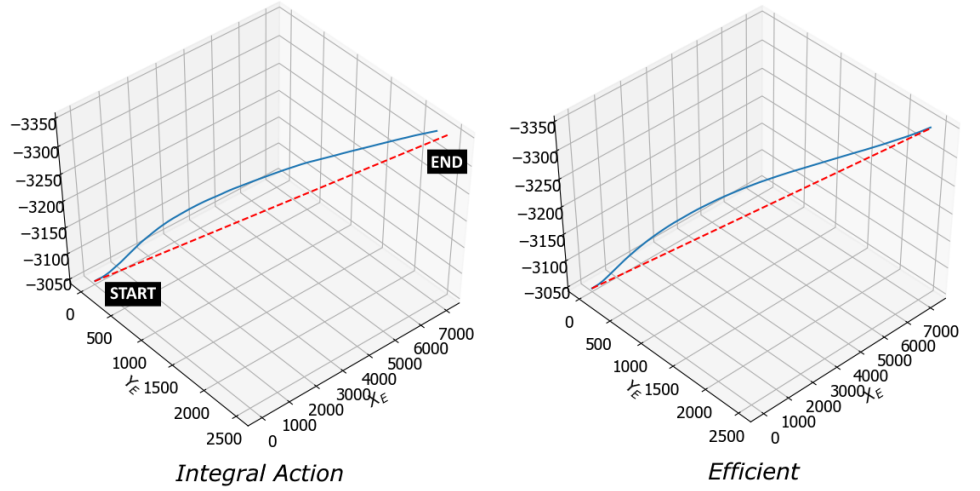

Figure 5.31: Single Waypoint - Tracking in the presence of faults

Firstly, the aircraft deviates largely in the Y plane before getting closer to the target for both MPC methods. This is because initially, $\psi_0 = 0$, so the aircraft has to also turn around 20 degrees while moving forward in X. To keep the MPC and QP stable, the roll angle ϕ was constrained to $\pm 0.5 rad$. Hence, given this condition the resulting trajectory is acceptable. From the above, the results are near identical for the fault conditions despite the faults occurring at $t = T/2$. This demonstrates that at the halfway point, the longitudinal and lateral setpoints are far from their maximum value, and hence the faults can be compensated for. To clearly illustrate, the CVs for the results in Fig. 5.31 are shown here.

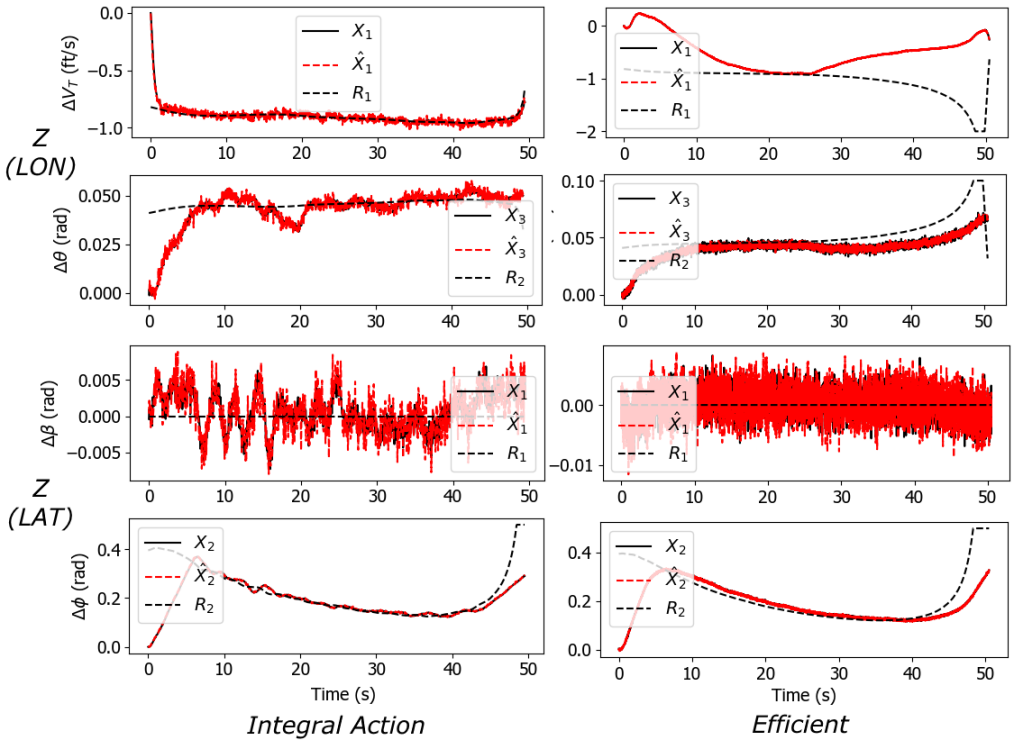

Figure 5.32: Control Variables - Tracking in the presence of faults

As can be seen, for ϕ, θ when $t \geq 20s$, their commanded values are around $0.05, 0.15 rad$ respectively, far from the limit of $\pm 0.1, \pm 0.5 rad$. In other words, the impact of the fault is minimal. This emphasizes the fact that for best MPC performance, the setpoints must be kept away from their limits as much as possible. Results for the semi helical flight trajectory (fault free and fault conditions) are shown in Figs. 5.33 and 5.34. Once again, the initial setting was $\psi_0 = 0$.

This time, the deviation in the Y position (ψ) is not as large at the beginning since the setpoint

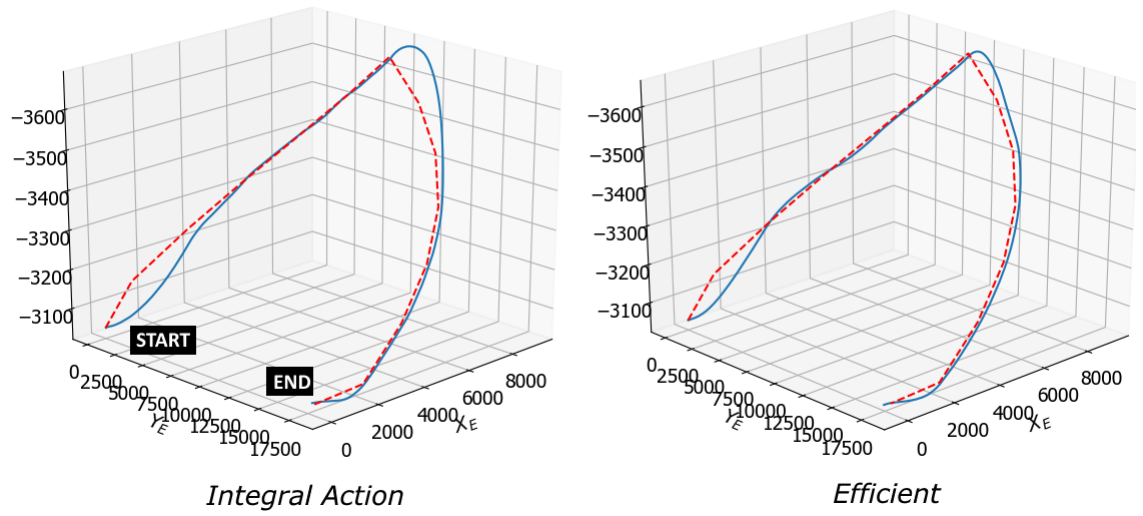

Figure 5.33: Helical Trajectory - Fault Free Tracking

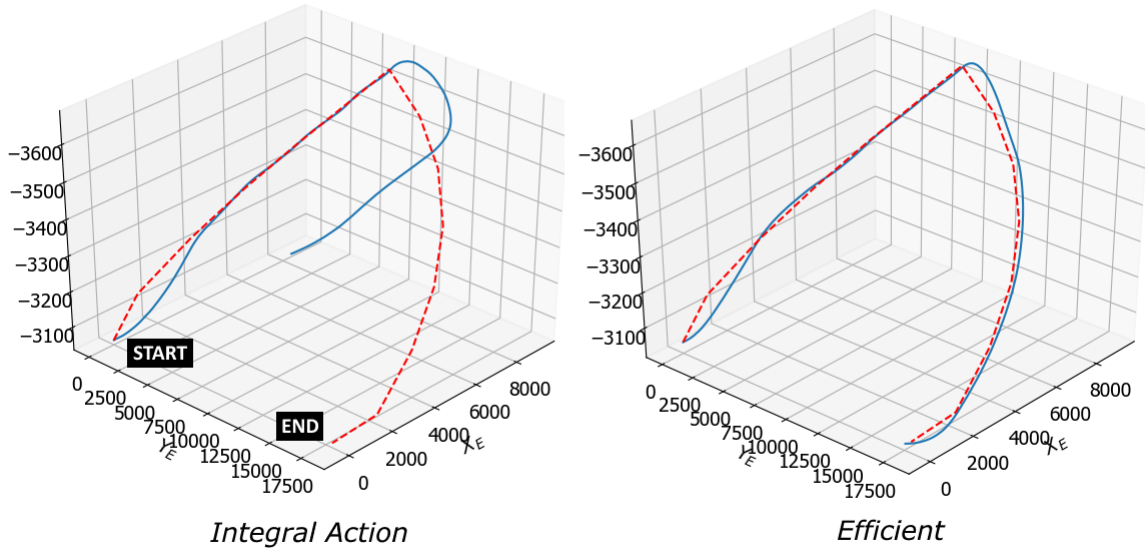

Figure 5.34: Helical Trajectory - Tracking in the presence of faults

change in ϕ small. By observing the results in Fig. 5.25, the reconfiguration mechanism of the efficient MPC method works very well here, as the tracking performace for a circular trajectory is orders of magnitude better than that for the integral action MPC, which fails to track the h and θ correctly. Furthermore, in general the efficient MPC keeps the setpoints away from their max and min limits. To emphasize this, Fig. 5.35 shows all CVs Z_{lon}, Z_{lat}. For the integral action MPC, θ is pushed to its maximum constraint ($-0.1rad$), and the fault in δ_e leads to a large steady state tracking error. This then significantly affects the resulting height, leading in the aircraft failing to track the required helix trajectory altitude. The reconfiguration mechanism plus the fact that the eMPC has a less overall steady state error in the presence of faults, leads to near perfect tracking of θ, leading to the aircraft tracking the helix correctly. Another point to note; the roll angle ϕ is not as severely affected from the fault in δ_a, which corresponds to the heading angle ψ being minimally affected throughout the flight, using both MPC methods.

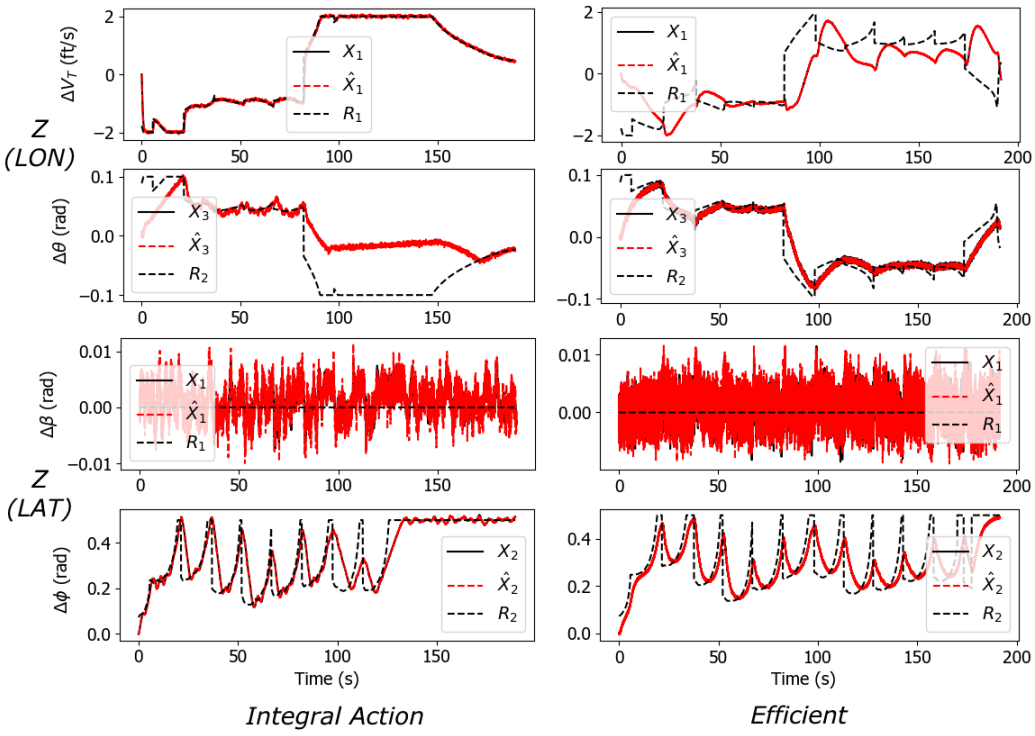

Figure 5.35: Control Variables - Tracking in the presence of faults

Fig. 5.36 shows all the inputs in the presence of faults. This is for the single waypoint tracking flight. The observer and QP work as expected with the constraint met, and correct estimation of

$\hat{U}_{f-lon}$ and $\hat{U}_{f-lat}$.

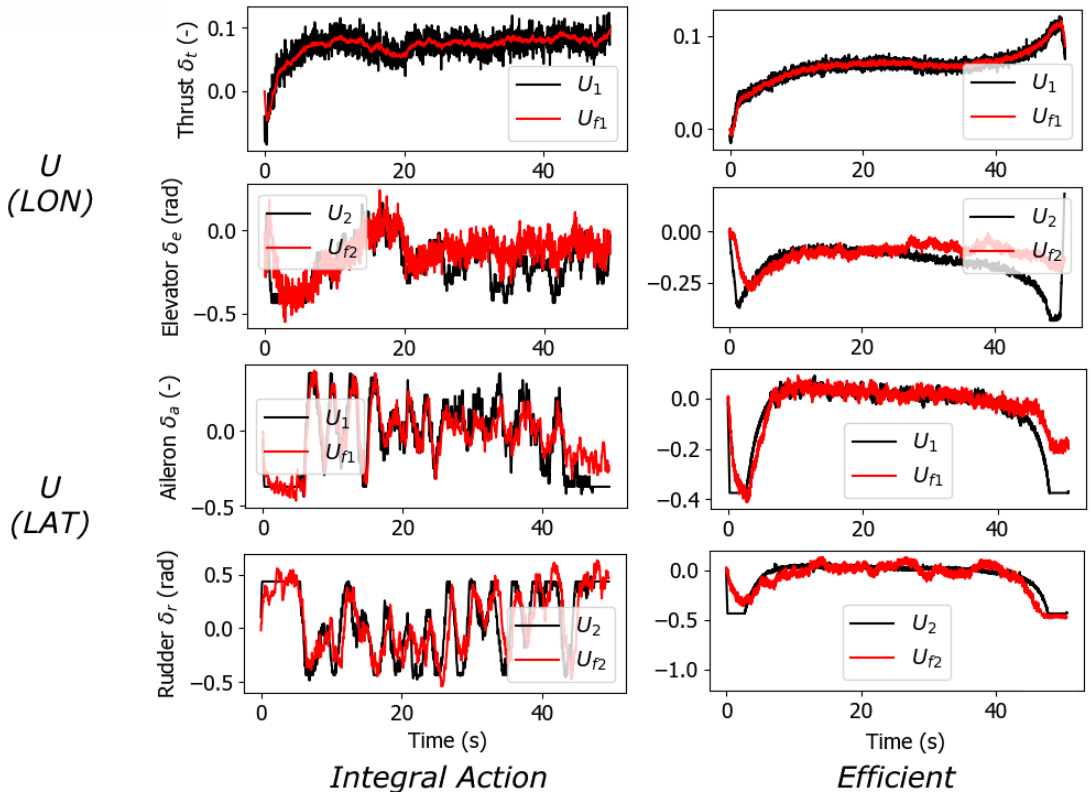

Figure 5.36: Inputs - Presence of faults

5.3.4 FlightGear Visualization

Lastly, a flight simulation is performed by using Matlab, Simulink, and FlightGear, a free open source flight simulator. The aircraft states and position / velocities were extracted from Python and imported into Matlab via a csv file. Then, a Simulink model was developed and connected to FlightGear. Specialized blocks within the Aerospace Toolbox in Matlab automatically convert position into latitude and longitude. Once that is done, a nice looking 3D animation can then be viewed by the user! Fig. 5.37 shows a portion of the block diagram in Simulink and Fig. 5.38 is the animation viewed in FlightGear.

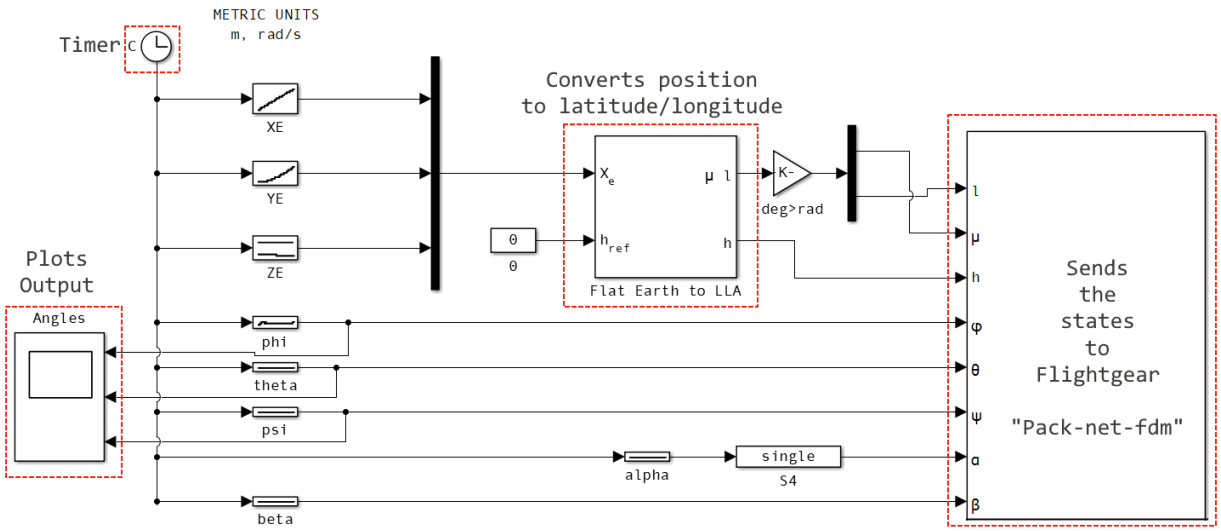

Figure 5.37: Simulink Block Diagram

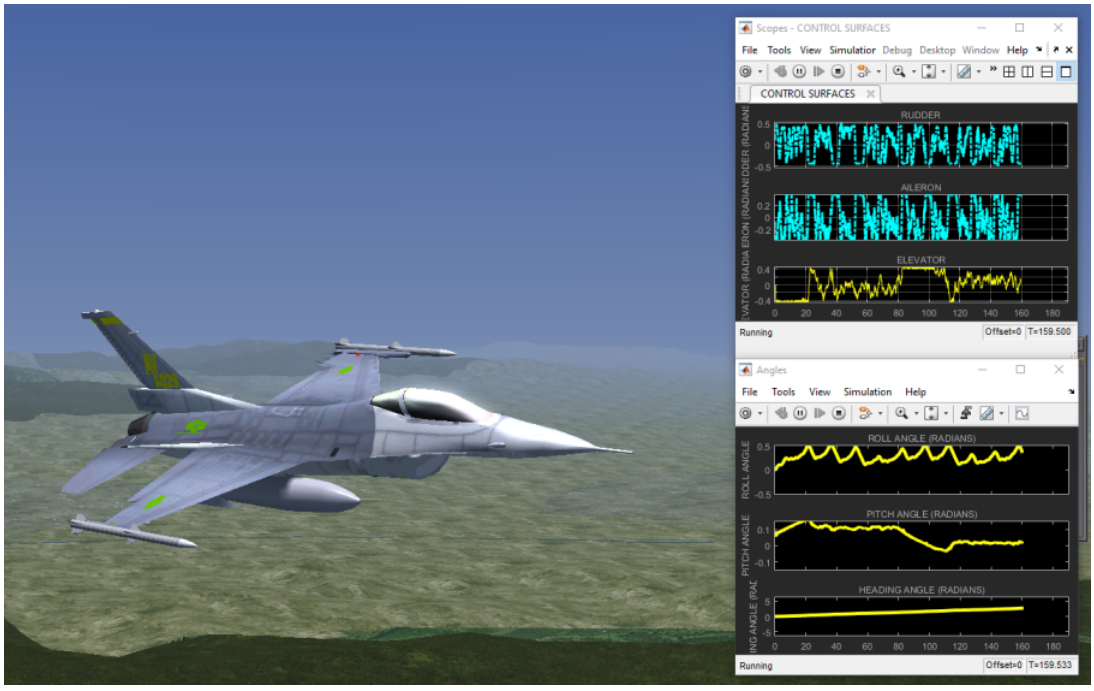

Figure 5.38: FlightGear Simulator

Chapter 6

Conclusion and Future Work

This study demonstrated the promising use of Model Predictive Control techniques with integrated Fault Detection, Diagnosis and Reconfiguration for multivariable systems with limited on-board capacity, such as aircraft and potentially systems like cars and boats, and builds upon the work performed by previous researchers at the Diagnosis, Flight Control and Simulation / Networked Autonomous Vehicles Lab at Concordia University. Due to the fact that real life operational constraints are always present on any system, it is key to develop the control system in a way that these limitations are considered, which then makes it very easy to then implement the algorithms onto the system motherboard, flight computer etc. This is the primary reason why Model Predictive Control is now making its way from the process control industry to other sectors such as aviation and self-driving cars. This study focused on applying the novel FDD based MPC methods for fixed-wing aircraft as they are a very good practical example of a constrained multivariable system with on-board computational limitations, prone to actuator faults. The open source F-16 models provided were of great use to test the developed MPC algorithms on.

Chapter 1 of this study provided a brief introduction to multivariable constrained systems in an engineering context. Chapter 2 described the two MPC algorithms (Efficient and Integral Action) in great detail. Chapter 3 provided the algorithms for the Fault Detection and Diagnosis method. It can be clearly seen that the observer based method proves superior to other more complex algorithms. It can be easily implemented on a microcontroller as part of the overall MPC architecture. The next two chapters provided the models of fixed wing flight dynamics and successful performance of

the proposed methods through simulations. The contributions of this study are highlighted through the simultaneous (on-line) use of the fault detection algorithms with MPC, and applying these algorithms on flight control design of fixed wing aircraft. The appendices contain detailed description of the novel constrained QP program along with the QP solvers used. Future potential work involves:

(1) Implementing these algorithms on hardware platforms to further study the performance and/or any extra limitations which may arise onboard.

(2) Using these methods for Linear Time Varying (LTV) systems, where the system matrices change in real time. This would require developing additional methods to ensure that on each time step, the closed loop MPC stays stable (both nominal and in the presence of faults).

(3) Implementing faster and more efficient QP algorithms specific to MPC purposes, knowing that Quadratic Programming is used in multiple engineering fields such as machine learning etc.

(4) Developing model reference adaptive MPC algorithms where if there is a sudden change in the system, the controller can react and match its nominal performance.

Appendix A

Constrained Quadratic Optimization

A.1 Overview

The field of optimization is widely used in engineering and finance, and it is the process for finding the solution of a variable which either maximizes or minimizes an objective (or cost) function, which may or may not be constrained. A simple example could be calculating the optimum amount of fuel required for maximum range whilst simultaneously minimizing the overall weight of the vehicle. In Quadratic Optimization, this cost function is quadratic and the general notation is:

$$(\underset{\mathbf{x}}{min}) f(x) = \frac{1}{2} x^T Q x + b^T x \longrightarrow Fx \leq G \tag{118}$$

In (118), the decision variable is a vector $x \in \mathbb{R}^n$, Q is a positive definite matrix ($Q \in \mathbb{R}^{n \times n}, Q > 0$), If there are p constraints, then they are represented in the form $Fx \leq G$, where $G \in \mathbb{R}^p$, contains the numerical values of the constraints, and F maps these constraints i.e. $F \in \mathbb{R}^{p \times n}$. Note that Q must be positive definite because only then will this objective function be convex, and a minimum solution can be found. An example of the constraint model is already demonstrated in 2.5.

A.2 QP Development - Efficient MPC

This section describes the novel constrained Quadratic Program developed for the efficient MPC method in [19]. Firstly, the cost function and the constraint model is provided within (2.3). To recap

the objective is to minimize the cost function:

$$(\underset{\mathbf{U}}{min})J = U^T HU + 2f^T U + E^T QE \longrightarrow MU < \gamma \tag{119}$$

The task is to now convert this cost function along with the constraints into a dual problem. This is done using the well known Primal Dual procedure, where the primal problem (above) is converted in to an equivalent Dual Problem in terms of the Lagrange Multiplier (λ). This is part of the Duality Principle, a key concept in optimization theory. To begin, using (119), the Lagrangian for the primal problem can be formed as:

$$\mathcal{L}(U, \lambda) = U^T HU + (2f^T + \lambda^T M)U + E^T QE - \lambda^T \gamma \tag{120}$$

Next, by setting $\partial\mathcal{L}/\partial U = 0$ and $\partial\mathcal{L}/\partial\lambda = 0$ the optimum solutions U^* and λ^* are obtained as:

$$U^* = -H^{-1}(f + 0.5M^T\lambda) \tag{121}$$

$$\lambda^* = -2[MH^{-1}M^T]^{-1}\gamma - 2[MH^{-1}M^T]^{-1}(MH^{-1})f \tag{122}$$

Then by substituting (121) into (120) and knowing the fact that $H = H^T \therefore H^{-1} = (H^{-1})^T = (H^T)^{-1}$, one can obtain:

$$\begin{aligned}(\underset{\mathbf{U}}{min})J = &-f^T H^{-1} f - f^T H^{-1} M^T \lambda \\ &- 0.25\lambda^T MH^{-1}M^T\lambda - \lambda^T\gamma + E^T QE \\ = &\frac{-1}{2}\lambda^T[0.5(MH^{-1}M^T)]\lambda \\ &- \lambda^T(MH^{-1}f + \gamma) - f^T H^{-1} f + E^T QE\end{aligned} \tag{123}$$

Next, defining a matrix $W \in \mathbb{R}^{(4m)\times(4m)}$ and a vector $Z \in \mathbb{R}^{4m}$ as:

$$W \triangleq 0.5(MH^{-1}M^T) \tag{124}$$

$$Z \triangleq MH^{-1}f + \gamma \tag{125}$$

The dual problem in terms of λ can then be written as:

$$(\underset{\lambda \geq 0}{max})q = \frac{-1}{2}\lambda^T W\lambda - \lambda^T Z - f^T H^{-1} f + E^T QE \tag{126}$$

This is now another QP problem with λ as the decision variable, and H, H^{-1} are both positive definite. The matrix W is positive semidefinite. This will be the case even if M does not have full column rank. Alternatively, the Dual Problem can also be expressed as:

$$(\underset{\lambda \geq 0}{min})q = \frac{1}{2}\lambda^T W\lambda + \lambda^T Z + f^T H^{-1} f - E^T QE \tag{127}$$

Here, $f^T H^{-1} f$ and $E^T QE$ are > 0.

It has been well established that the dual problem is a convex optimization problem regardless of the convexity of the primal problem. In order to ensure that the primal and dual problem are equivalent, setting $\partial q/\partial\lambda = 0$, results in the optimum solution as: $\lambda^* = -W^{-1}\gamma - W^{-1}(MH^{-1})f$, and after substituting (124) and (125), this equals the optimum λ^* obtained for the primal problem, i.e. (122).

Remark : The scalars $E^T QE$ and $f^T H^{-1} f$ can be dropped without loss of generality, since they do not have an impact on the optimal solutions for U and λ.

A.3 QP Program - Integral Action MPC

This section describes the Quadratic Program for the Integral-Action MPC method. In the literature, the cost function is slightly different, and the primal dual procedure is provided in detail in [21]. However for this study, the cost function was changed to match with the previous method. Hence, the Primal Dual procedure for this is almost identical, and will be highlighted here. Note that a slight change of the cost function does not affect the end result, i.e as long as $\mathcal{H}$ is positive definite and the Dual problem is equivalent to the Primal problem, a solution can be found. Firstly, as a recap from 30, the objective is to:

$$(\underset{\Delta U}{min})J = \Delta U^T \mathcal{H}\Delta U + 2\bar{f}^T U + E^T \bar{Q}E \longrightarrow \bar{M}\Delta U \leq \bar{\gamma} \tag{128}$$

Going through the same steps from (120) to (123), the terms for W and Z can be obtained, this time they will be called $\bar{W}$ and $\bar{Z}$ to differentiate. Here, $\bar{W} = \mathbb{R}^{(4m \cdot N_c) \times (4m \cdot N_c)}$ and $\bar{Z} = \mathbb{R}^{4m \cdot N_c}$.

$$\bar{W} \triangleq 0.5(\bar{M}\mathcal{H}^{-1}\bar{M}^T) \tag{129}$$

$$\bar{Z} \triangleq \bar{M}\mathcal{H}^{-1}\bar{f} + \bar{\gamma} \tag{130}$$

The Dual Problem then becomes:

$$\underset{\lambda \geq 0}{(min)} q = \frac{1}{2}\lambda^T \bar{W}\lambda + \lambda^T \bar{Z} + \bar{f}^T \mathcal{H}^{-1}\bar{f} - E^T \bar{Q} E \tag{131}$$

A.4 Activating a QP Solver

Once the Dual Problem is obtained (in terms of λ), the objective is to now solve for the optimum λ, and then the optimal input U or ΔU can be calculated. A graphical representation is shown in A.1. Hence, the terms $W/\bar{W}$ can be defined beforehand. However since $\gamma/\bar{\gamma}$ is updated on each time step, $Z/\bar{Z}$ has to be updated as well. Note that $dim(\lambda) = dim(\gamma) = dim(\bar{\gamma})$

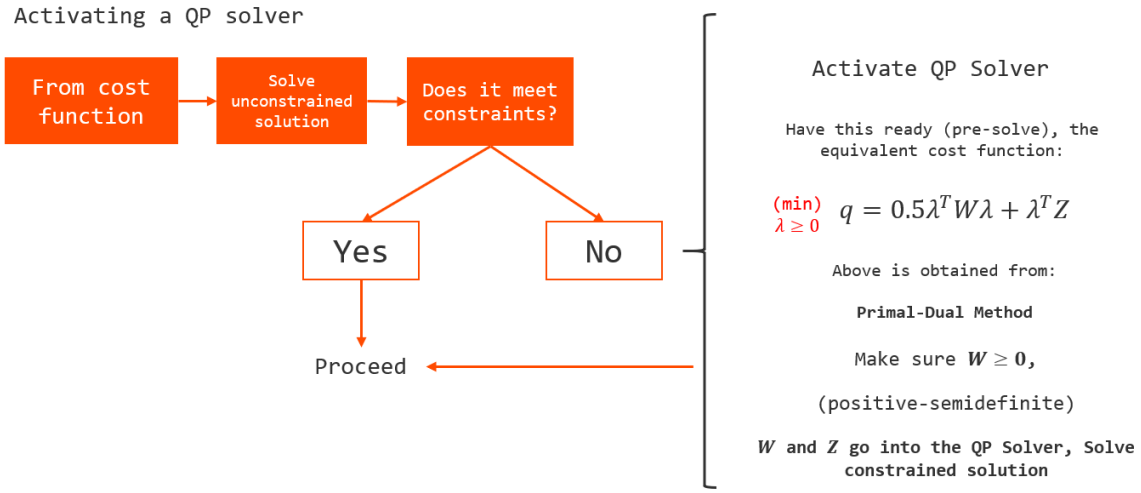

Figure A.1: Activating the Quadratic Programming Solver - Flowchart

Appendix B

Quadratic Programming Solvers

B.1 Basic Information

From the Dual Problem, the goal is to now solve for the optimum Lagrange multiplier λ^*. The generic algorithm for a QP solver works as follows. For convenience, W, Z, U will be shown, however it's the identical process for $\bar{W}, \bar{Z}, \Delta U$:

Algorithm 3 Generic QP Solver

1: **if** $MU_u > \gamma$ **then**
Require: W, Z
Require: $v = 0$ ▷ Number of Iterations
Require: $\lambda_{v=0}, e_{v=0}$ ▷ Initial Lagrange multiplier and error
Require: $c \simeq 0, c > 0$ ▷ Error Convergence term
Ensure: $e_{v=0} > 0.1$ ▷ Initial error - program stops when $e \simeq 0$
2: **while** $e_v > c$ **do**
3: Get λ_{v+1}
4: e = $\lambda_{v+1} - \lambda_j$
5: $v = v + 1$ ▷ Update λ, error term and iterations
6: **end while**
7: **return** $\lambda_v \leftarrow \lambda^*$ ▷ This is the optimal λ
8: **end if**
9: Get U_c from λ_v ▷ Get the optimal input

Algorithm 3 provides the generic model of a QP Solver. The error convergence term c should be set very close to zero. For instance $c = 1e - 6$ works well. It should not be set too low otherwise the number of iterations required will increase and there will be more delay. During each iteration

of the solver loop, the λ is calculated and error is compared. Eventually, provided that $W \geq 0$ and the problem is feasible the QP converges where $(\lambda_{v+1} - \lambda_v) \leq c$. There are several algorithms to calculate λ, however this study focused on the Hildeth's Quadratic Programming Procedure and the Parallel Quadratic Programming Procedure. In both these methods, each component $\lambda_i,\ \forall i \in dim(\lambda)$ is solved for. Note that $dim(\lambda) = dim(Z)$ or $dim(\bar{Z})$. This leads to the conclusion that for the efficient MPC, $max(dim(Z)) = 4m$ since the control horizon is unity. However for the integral action MPC, $max(dim(\bar{Z})) = 4m \cdot N_c$, and since $N_c > 1$, it would take several times longer for the QP solver to obtain the optimum λ. This is why the efficient MPC algorithm is so useful for aircraft as the time delay is minimized!

B.2 Hildreth's Quadratic Programming Method

For convenience, W, Z will be shown, however it's the same idea for $\bar{W}, \bar{Z}$.

The Hildreth's QP procedure is a well known method and has been used for decades [12]. It relies on active set methods to get the optimum λ. During each step, λ is varied one at a time and is adjusted to minimize the objective function (i.e the Dual Problem). However, if the value of λ needed to minimize the Dual Problem is ≤ 0, then that component of λ is set to zero. The HQP method relies on an element-by-element search and does not require matrix inversion. The optimum λ vector contains $\lambda_i = 0$ for active constraints, and $\lambda_i > 0$ for inactive constraints. Considering one complete cycle (v is the number of iterations), the method is:

$$\lambda_i^{v+1} = max(0, w_i^{v+1}) \tag{132}$$

$$w_i^{v+1} \rightarrow -\frac{1}{h_{ii}}\Big[k_i + \sum_{j=1}^{i-1} h_{ij}\lambda_j^{v+1} + \sum_{j=i+1}^{n} h_{ij}\lambda_j^{v}\Big] \tag{133}$$

In (133), the terms h_{ij}, k_i are given as:

$$h_{i,j} = W_{i,j} \tag{134}$$

$$k_i = Z_i \tag{135}$$

Notice how $w_i^{v+1} = f(\lambda_j^v, \lambda_j^{v+1})$, i.e it depends on λ at the current and the next iteration. The strong benefit of the HQP algorithm is that it takes a very small amount of iterations to converge as shown in 5.1.1, however it does involve a maximum search which may take time especially for higher order systems, or if $dim(\lambda)$ is very large.

The HQP algorithm is presented in Algorithm 4:

Algorithm 4 HQP Solver

Require: W, Z
Require: $r = dim(Z)$
Require: $\lambda = 0_r, e > 0$
while $e > 1e-6$ **do**
  $\lambda_o = deepcopy(\lambda)$
  for $i = r$ **do**
    $w = W[i]\lambda - W_{i,i}\lambda_i + Z_i$
    $w = -w/W_{i,i}$
    $\lambda_i = max(0, w)$
  end for
  $e = (\lambda - \lambda_o)^T(\lambda - \lambda_o)$
end while
$\lambda^* = \lambda$
return λ^*

For a Quadratic Programming solver, it is best to use the simplest solver which provides the solution in the fastest amount of time. For real time embedded applications such as control theory and MPC, the Parallel Quadratic Programming method is superior, and will be explained next.

B.3 Parallel Quadratic Programming Method

The Parallel Quadratic Programming (PQP) procedure was first developed at Mitsubishi Electric Research Laboratories (MERL). Instead of a maximum search, this algorithm uses a simple iterative multiplication process to get the optimum λ. This means that only a single equation is needed, and several terms can be defined beforehand to save time. The update law is given by:

$$\lambda_i = \lambda_i \left[\frac{Z_i^- + (W^-\lambda)_i}{Z_i^+ + (W^+\lambda)_i} \right] \tag{136}$$

The PQP algorithm is presented here as follows: Note that the max operator is taken elementwise.

Algorithm 5 PQP Solver

Require: $Z_i^- = max(-Z, 0), Z_i^+ = max(Z, 0)$
Require: $r = 0_{dim(Z)}$
Require: $W^- = max(-W, 0) + diag(r)$
Require: $W^+ = max(W, 0) + diag(r)$
Require: $\lambda > 0, e > 0$
while $e > 1e - 6$ **do**
    $\lambda_o = deepcopy(\lambda)$
    for $i = 1 : dim(r)$ **do**
        $\lambda_i = \lambda_i \left[\frac{Z_i^- + (W^-\lambda)_i}{Z_i^+ + (W^+\lambda)_i} \right]$
    end for
    $e = (\lambda - \lambda_o)^T(\lambda - \lambda_o)$
end while
$\lambda^* = \lambda$
return λ^*

The vector r is chosen such that $\forall i, W_{ii} + r_i > 0$. Choosing the smallest r possible reduces the number of iterations for convergence. In order for convergence, the initial guess value for the PQP algorithm must be $\lambda^{j=0} > 0$.

B.4 HQP vs PQP - Performance

Fig. B.1 shows the number of iterations of the HQP vs PQP algorithm to converge.

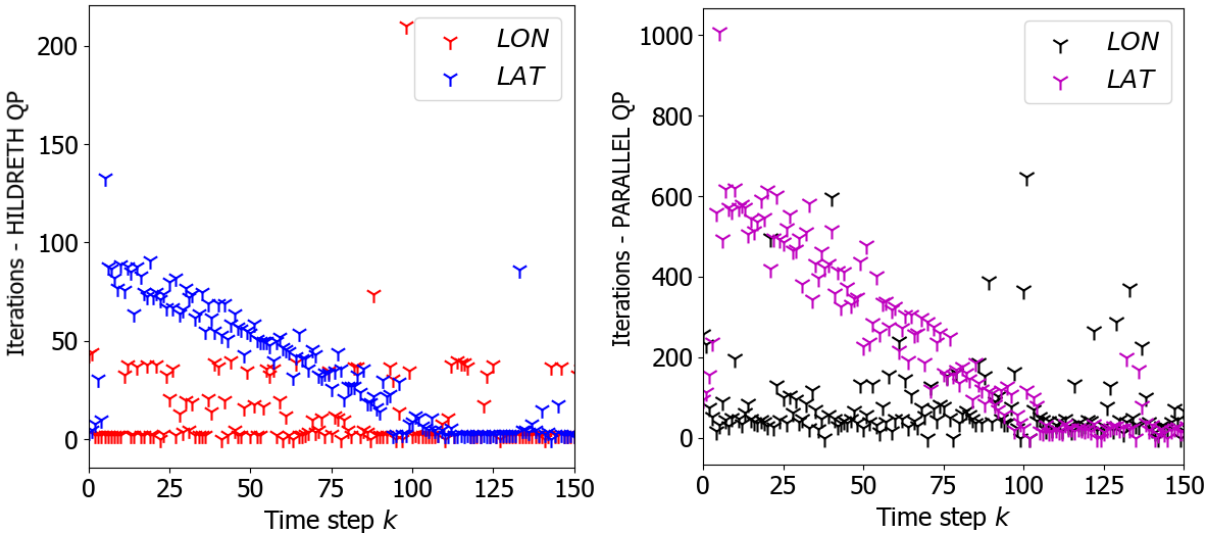

Figure B.1: Number of Iterations - HQP (left) vs PQP (right)

The iterations for Fig. B.1 were taken from the single waypoint trajectory simulation described in 5.3.3. In addition, it was for the integral action MPC algorithm without faults present. The following conclusions can hence be drawn:

- As the setpoints for the heading and roll angles (ψ_{cmd}, ϕ_{cmd}) were large, this resulted in a large error (E), hence it intuitively took longer for the lateral MPC controller to calculate the optimal constrained ΔU_{lat} for both the HQP and PQP cases.
- Eventually, as $\psi_{cmd} - \hat{\psi} \rightarrow 0$, i.e the error approaching zero, the number of iterations reduced from $100, 600$ respectively to zero at $k \simeq 100$ or $t \simeq 100 \cdot 0.05 = 2.0s$
- The longitudinal setpoint for θ, V_T was not large (it's constrained at $|\theta| \leq 0.1rad$. Hence, it does not take a lot of iterations for the longitudinal MPC controller calculate the optimal constrained ΔU_{lon}.
- It takes approximately 5.5-6 times more iterations for the PQP vs. the HQP algorithm.

Future work will be required to experimentally verify the overall time taken for each of these algorithms when they are implemented in low level format, on hardware. Conclusively, both algorithms produce the same result and are very stable, thus making it ideal for real time applications including but not limited to Model Predictive Control.

Bibliography

[1] M. V. Cook, *Flight Dynamics Principles: A Linear Systems Approach to Aircraft Stability and Control*. Butterworth-Heinemann, 2012.

[2] C. C. Wo and Z. Q. Min, "Coupling & decoupling control study on aircraft (Airbus A320)," in *2016 8th International Conference on Modelling, Identification and Control (ICMIC)*. IEEE, 2016, pp. 29–36.

[3] S. Snell and R. Hess, "Robust, decoupled, flight control design with rate-saturating actuators," *Journal of Guidance, Control, and Dynamics*, vol. 21, no. 3, pp. 361–367, 1998.

[4] P. E. Pounds, D. R. Bersak, and A. M. Dollar, "Stability of small-scale UAV helicopters and quadrotors with added payload mass under PID control," *Autonomous Robots*, vol. 33, no. 1-2, pp. 129–142, 2012.

[5] B. Godbolt, N. I. Vitzilaios, and A. F. Lynch, "Experimental validation of a helicopter autopilot design using model-based PID control," *Journal of Intelligent & Robotic Systems*, vol. 70, no. 1-4, pp. 385–399, 2013.

[6] L. Sonneveldt, Q. Chu, and J. Mulder, "Nonlinear flight control design using constrained adaptive backstepping," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 2, pp. 322–336, 2007.

[7] S. Seshagiri and E. Promtun, "Sliding mode control of F-16 longitudinal dynamics," in *2008 American Control Conference*. IEEE, 2008, pp. 1770–1775.

[8] K. R. Muske and J. B. Rawlings, "Model predictive control with linear models," *AIChE Journal*, vol. 39, no. 2, pp. 262–287, 1993.

[9] D. L. Di Ruscio, "Model predictive control with integral action: A simple MPC algorithm," *Norwegian Society of Automatic Control*, 2013.

[10] P. W. Gibbens and E. D. Medagoda, "Efficient model predictive control algorithm for aircraft," *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 6, pp. 1909–1915, 2011.

[11] S. Sun, L. Dong, L. Li, and S. Gu, "Fault-tolerant control for constrained linear systems based on MPC and FDI," *International Journal of Information and Systems Sciences*, vol. 4, no. 4, pp. 512–523, 2008.

[12] C. Hildreth, "A quadratic programming procedure," *Naval Research Logistics Quarterly*, vol. 4, no. 1, pp. 79–85, 1957.

[13] M. Brand, V. Shilpiekandula, C. Yao, S. A. Bortoff, T. Nishiyama, S. Yoshikawa, and T. Iwasaki, "A parallel quadratic programming algorithm for model predictive control," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 1031–1039, 2011.

[14] Y. M. Zhang and J. Jiang, "Bibliographical review on reconfigurable fault-tolerant control systems," *Annual Reviews in Control*, vol. 32, no. 2, pp. 229–252, 2008.

[15] N. E. Wu, Y. M. Zhang, and K. Zhou, "Detection, estimation, and accommodation of loss of control effectiveness," *International Journal of Adaptive Control and Signal Processing*, vol. 14, no. 7, pp. 775–795, 2000.

[16] W. Chen and M. Saif, "Observer-based fault diagnosis of satellite systems subject to time-varying thruster faults," *ASME Journal of Dynamic Systems, Measurement and Control*, vol. 129, no. 3, pp. 352–356, 2007.

[17] E. D. Medagoda and P. W. Gibbens, "Multiple horizon model predictive flight control," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 3, pp. 946–951, 2014.

[18] E. Medagoda and P. Gibbens, "Efficient predictive flight control," in *ICCAS 2010*. IEEE, 2010, pp. 1297–1302.

[19] V. Deshpande and Y. Zhang, "Multivariable receding horizon control of aircraft with actuator constraints," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2020, pp. 1846–1851.

[20] Z. Zheng, Z. Wang, J. Zhao, and H. Zheng, "Constrained model predictive control algorithm for cascaded irrigation canals," *Journal of Irrigation and Drainage Engineering*, vol. 145, no. 6, p. 04019009, 2019.

[21] L. Wang, "Model predictive control: Design and implementation using MATLAB (T-3)," in *2009 American Control Conference*. IEEE, 2009, pp. 25–26.

[22] G. Pannocchia and J. B. Rawlings, "Disturbance models for offset-free model-predictive control," *AIChE journal*, vol. 49, no. 2, pp. 426–437, 2003.

[23] R. Mehra, "On the identification of variances and adaptive Kalman filtering," *IEEE Transactions on Automatic Control*, vol. 15, no. 2, pp. 175–184, 1970.

[24] M. H. Amoozgar, A. Chamseddine, and Y. M. Zhang, "Experimental test of a two-stage Kalman filter for actuator fault detection and diagnosis of an unmanned quadrotor helicopter," *Journal of Intelligent & Robotic Systems*, vol. 70, no. 1-4, pp. 107–117, 2013.

[25] Y. Zhong, W. Zhang, Y. M. Zhang, J. Zuo, and H. Zhan, "Sensor fault detection and diagnosis for an unmanned quadrotor helicopter," *Journal of Intelligent & Robotic Systems*, vol. 96, no. 3, pp. 555–572, 2019.

[26] Y. Zhong, Y. M. Zhang, W. Zhang, J. Zuo, and H. Zhan, "Robust actuator fault detection and diagnosis for a quadrotor UAV with external disturbances," *IEEE Access*, vol. 6, pp. 48 169–48 180, 2018.

[27] C. Edwards and S. K. Spurgeon, "On the development of discontinuous observers," *International Journal of Control*, vol. 59, no. 5, pp. 1211–1229, 1994.

[28] B. L. Walcott and S. H. Zak, "Combined observer-controller synthesis for uncertain dynamical systems with applications," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 18, no. 1, pp. 88–104, 1988.

[29] W. Chen and M. Saif, "An actuator fault isolation strategy for linear and nonlinear systems," in *Proceedings of the 2005, American Control Conference, 2005.* IEEE, 2005, pp. 3321–3326.

[30] W. Morse and K. Ossman, "Model following reconfigurable flight control system for the AFTI/F-16," *Journal of Guidance, Control, and Dynamics*, vol. 13, no. 6, pp. 969–976, 1990.

[31] L. Sonneveldt, "Nonlinear F-16 model description," *Delft University of Technology - Course A3M35SRL*, 2006.

[32] J. Leonard, "Investigation of lateral-directional coupling in the longitudinal responses of a transfer function simulation model," Ph.D. dissertation, Virginia Tech, 2003.