DATA-DRIVEN MODEL DISCOVERY AND CONTROL OF LATERAL-DIRECTIONAL FIGHTER AIRCRAFT DYNAMICS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF APPLIED MATHEMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

CAN ÖZNURLU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
SCIENTIFIC COMPUTING

AUGUST 2022

Approval of the thesis:

DATA-DRIVEN MODEL DISCOVERY AND CONTROL OF LATERAL-DIRECTIONAL FIGHTER AIRCRAFT DYNAMICS

submitted by **CAN ÖZNURLU** in partial fulfillment of the requirements for the degree of **Master of Science in Scientific Computing Department, Middle East Technical University** by,

Prof. Dr. A. Sevtap Kestel _____
Dean, Graduate School of **Applied Mathematics**

Assoc. Prof. Dr. Önder Türk _____
Head of Department, **Scientific Computing**

Prof. Dr. Ömür Uğur _____
Supervisor, **Scientific Computing, METU**

Assoc. Prof. Dr. Tayfun Çimen _____
Co-supervisor, **Turkish Aerospace**

Examining Committee Members:

Prof. Dr. Metin Uymaz Salamcı _____
Mechanical Engineering, Gazi University

Prof. Dr. Ömür Uğur _____
Scientific Computing, METU

Assist. Prof. Dr. Ali Türker Kutay _____
Aerospace Engineering, METU

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: CAN ÖZNURLU

Signature :

ABSTRACT

DATA-DRIVEN MODEL DISCOVERY AND CONTROL OF
LATERAL-DIRECTIONAL FIGHTER AIRCRAFT DYNAMICS

Öznurlu, Can
M.S., Department of Scientific Computing
Supervisor : Prof. Dr. Ömür Uğur
Co-Supervisor : Assoc. Prof. Dr. Tayfun Çimen

August 2022, 83 pages

The focus of this thesis is to control the lateral-directional motion of the fighter aircraft by using integral action based Model Predictive Control (MPC) where the model is obtained by data-driven model discovery method. Dynamic Mode Decomposition with Control (DMDc) is used as a model discovery technique based only on measurement data with no modeling assumptions. The model created using this technique is used for MPC and tested against noisy conditions. In addition, performance comparison of MPC with Classical Controller is carried out. Finally, Speedgoat Unit Real-Time Target Machine®, which offers a real-time testing is used to verify the generated DMDc-MPC algorithm and understand the computational cost.

The results show that the DMDc model discovery method performs very well in noise-free situations and meets the evaluation criteria together with MPC. However, its performance decreases in the presence of measurement noise. Finally, real-time test results on Speedgoat® equipment have shown that the generated DMDc-MPC algorithm has low computational cost and can be used in systems with low computational power.

Keywords: DMDc, Modelling, System Identification, Data-Driven Control, Model Predictive Control, Fighter Aircraft Dynamics

ÖZ

YANAL VE YÖNLÜ SAVAŞ UÇAĞI DİNAMİKLERİNİN VERİ TABANLI YÖNTEMLER İLE MODEL KEŞFİ VE KONTROLÜ

Öznurlu, Can
Yüksek Lisans, Bilimsel Hesaplama Bölümü
Tez Yöneticisi : Prof. Dr. Ömür Uğur
Ortak Tez Yöneticisi : Doç. Dr. Tayfun Çimen

Ağustos 2022, 83 sayfa

Bu çalışma, veri tabanlı model keşfedimi teknikleri kullanılarak oluşturulan uçak modelinin, model öngörülü kontrol (MPC) ile integral aksiyonu için kullanılarak savaş uçağının yanal ve yönlü hareketinin kontrolüne odaklanmaktadır. Sadece zamana bağlı ölçümlere dayalı model keşfi için DMDc regresyon tekniği kullanılmıştır. Bu teknik kullanılarak oluşturulan model, MPC için kullanılmış ve gürültülü durumlara karşı test edilmiştir. Ayrıca MPC'nin Klasik Kontrolör ile performans karşılaştırması yapılmıştır. Son olarak, gerçek zamanlı test imkanı sunan Speedgoat Unit Real-Time Target Machine®, üretilen DMDc-MPC algoritmasını doğrulamak ve hesaplama maliyetini anlamak için kullanıldı.

Sonuçlar, DMDc model keşif yönteminin gürültüsüz durumlarda çok iyi performans gösterdiğini ve MPC ile birlikte değerlendirme kriterlerini karşıladığını göstermektedir. Fakat ölçüm gürültüsü varlığında performansında düşüş göstermektedir. Son olarak, Speedgoat® ekipmanı üzerindeki gerçek zamanlı test sonuçları üretilen DMDc-MPC algoritmasının hesaplama maliyetinin düşük olduğunu ve hesaplama gücü düşük olan sistemlerde kullanılabileceğini göstermiştir.

Anahtar Kelimeler: DMDc, Modelleme, Sistem Tanımlaması, Veri Tabanlı Kontrol, Model Öngörülü Kontrol, Savaş Uçağı Dinamikleri

To all my loved ones

ACKNOWLEDGMENTS

Throughout the writing of this dissertation I have received a great deal of support and assistance.

First of all, I am grateful to my supervisor Prof. Dr. Ömür Uğur for accepting me as a master student and giving me the opportunity and support to write my thesis in this field.

Assoc. Prof. Dr. Tayfun Çimen taught me not only the essential language of Control, but he has been, for many years, my most cherished conversation partner in that magical language, in which we have questioned by our beloved aircraft, TF-X.

I am grateful to Dr. Murat Millidere for leading me into the applied mathematics and optimization field in the first place.

I would like to thank my colleagues and friends Mustafa Yüce and Berke Bayrı for their support. This process would have been more difficult without their funny jokes and uplifting conversations.

I also thank to Esra Halaçlar who has boosted me with positive energy in all periods of my university life.

In my pursuit of knowledge I have been most profoundly influenced by following teachers throughout my career: Mr. Cahit Çıray and Mr. İlkay Yavrucuk. My interest and motivation in aerospace is increased thanks to their enthusiasm and vision.

This thesis is supported by Turkish Aerospace. This place has been like a second home for me. I am very grateful to this place which provides me with a good working environment and the opportunity to improve myself.

Last, but not least, I owe nearly all I have accomplished to my mother. Above all else, she has provided me with a home and family, which I consider to be the most valuable things I own.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CFD	Computational fluid dynamics
CLAW	Control law
CPU	Central processing unit
DM	Delay margin
DDC	Data driven control
DMD	Dynamic mode decomposition
DMDc	Dynamic mode decomposition with control
DOF	Degree of freedom
GM	Gain margin
GPU	Graphics processing unit
ILC	Iterative learning control
IFT	Iterative feedback tuning
ISA	International Standard Atmosphere
LQR	Linear quadratic regulator
MBC	Model based control
MFAC	Model-free adaptive control
MIMO	Multiple-input and Multiple-output
MPC	Model predictive control
NN	Neural network
QP	Quadratic programming
TET	Task execution time
SINDy	The sparse identification of nonlinear dynamics
SVD	Singular value decomposition
VRFT	Virtual reference feedback tuning

CHAPTER 1

INTRODUCTION

For many systems, the main goal is to actively manipulate the behavior of the system in line with a given engineering objective. Manipulating the behavior of a system to achieve a desired goal is commonly known as control theory and is one of the most successful areas at the intersection of applied mathematics and engineering. Control theory is highly related to data science, as it uses sensor measurements (data) to achieve the given goal.

Control system consists of two main parts, the controlled object and the controller. Control theory can be divided into two main headings, namely model-based control (MBC) theory and data-driven control (DDC) theory. Real-world controllable objects can be studied in four classes:

- Those for which precise mathematical models based on the identification or first principles are accessible.
- Those for which mathematical models based on identification or first principles are roughly correct with modest uncertainty.
- Those for which first principles or identification-based mathematical models are complicated with too high order and too much nonlinearity, etc.
- Those for which it is difficult or impossible to build first principles or identification-based mathematical models.

These classes and their relations with each other is shown in Figure 1.1 [18].

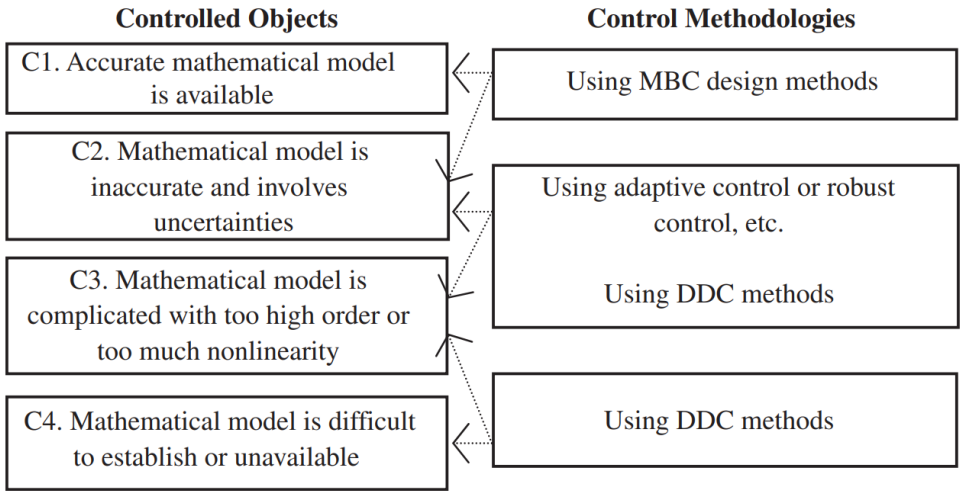


Figure 1.1: Controlled objects and control methodologies [18].
Reprinted from *"From model-based control to data-driven control: Survey, classification and perspective"* by Hou and Wang, 2013, Information Sciences, 235, 3-35, Copyright by Elsevier

Only in the presence of a solid mathematical model MBC theory can provide solutions to issues. In addition, the uncertainties in this mathematical model must be within a certain limit. As is observed in Figure 1.1, we can deduce that only C1 and C2 classes are engaged in MBC. Naturally, the first question that comes to mind is what a solution to the problems in the C3 and C4 classes can be. DDC methods are the inevitable choice for these classes. Based on observation, the best control method should be able to provide solutions to all four classes of controlled objects.

1.1 Motivation

The most challenging, time-consuming and demanding task for modeling, simulation and control engineers in control projects is to write a mathematical model of the object to be controlled. This challenge can be overcomed by using data-driven model discovery methods such as many modern technniques in machine learning (e.g. neural networks (NN) [11]), sparse identification of nonlinear dynamics (SINDy) [42] and the autoregressive models [2] (e.g. ARX, ARMA, NARX, and NARMAX) that build models based entirely on the data collected. Later, control engineers can use this discovered model for control. However, the hard part is to come up with a simple but reliable model that reflects the dynamics of the system. In addition, if this system is

to work in real time, the model discovery and control algorithm presented should not be computationally expensive because systems with weak computational power may not handle such a load.

Therefore, the main motivation in this thesis is to come up with a simple but reliable model discovery and control working structure that can be applied in real time.

1.2 Literature Survey

Modern control theory has grown and made significant progress since the late 1960s. System identification, linear control, adaptive control, robust control and optimal control, which are branches of modern control theory, are frequently used in industrial processes, especially in the field of aerospace.

Modern control theory, often known as MBC, was born with the introduction of the parametric state space model by Kalman [24]. There are many successful applications in the aerospace industry where there are models with high precision (e.g. the flight control system of F-16 aircraft [39]). Top-down physics-based methodologies are used in traditional aircraft design and control procedures. However, construction of physics-based models or identification of corresponding parameters within the model (e.g. fighter aircraft dynamics at high angle of attack) may be problematic due to complicated, uncertain, and noisy working conditions [10].

One of the assumptions of MBC theory is the certainty equivalence principle. If there is a significant mismatch between the plant model and the established model, MBC design might not perform properly. Therefore, if the controller is designed with an unrealistic model, it will result in either poor performance or an unstable system in its closed loop. Minor modeling errors can lead to poor closed loop controller performance [50].

Practical operations in the chemical industry, metallurgy, machinery, electronics, power, transportation, and logistics have all changed dramatically as a result of the advancement of information science and technology. Manufacturing technologies and equip-

ment are used on a wide scale in these industries, and production processes have gotten increasingly sophisticated. It has gotten more difficult to model processes using first principles or identification. As a result, standard MBC theory is no longer applicable to control challenges in these types of business. Furthermore, many industrial processes generate and store massive volumes of processed data at all times of the day, containing all of the important state information about process operations and equipments. Using these data to directly build controllers, anticipate and assess system states, evaluate performance, make decisions, or even detect errors, both on-line and off-line, would be extremely beneficial, especially given the lack of precise process models. Therefore, both in theory and in practice, the establishment and growth of DDC are critical challenges.

It was computer science that first coined the phrase 'data-driven". The term is named by Johns (1936-2009) who pioneered data-driven learning. Initially, it was in an article, *"Should you be persuaded: Two examples of data-driven learning (1991)"* [22]. It began to appear in the control community's vocabulary since the mid-90s [21]. Up until now, there have been a number of DDC techniques, but they go by a variety of names. Data-based control, MFAC (model-free adaptive control) [17], IFT (iterative feedback tuning) [16], VRFT (virtual reference feedback tuning) [15], and ILC (iterative learning control) [55] are some of the DDC methods that have been developed thus far.

Although DDC-based studies are still considered as new, they have received great attention from the control community. Among the DDC approaches, machine learning methods, especially neural network-based ones, are quite common. Model discovery and control of an aerobatic helicopter [44], quadcopter [58] and aircrafts [9] are studied using neural networks (NNs). Despite the undeniably successful applications of NN, its need for huge datasets for the training process and the lack of guarantee that it can work outside the area where the data were collected makes its usefulness questionable. The difficulties in data-driven discovery for real-time control of nonlinear, high-dimensional, high-scale systems limit the use of NNs. Since it cannot respond quickly to sudden changes, it is difficult to use in online practical applications [23]. As a result, the solution is typically computationally heavy because of

information-rich nature of NN implementation. This requires the use of more powerful computational devices (e.g., a GPU) [10].

One of the MBC method, model predictive control (MPC) is a cornerstone of advanced process control, and is well-positioned to take advantage of the data-driven revolution. MPC is ubiquitous, especially in industrial applications, as it enables the highly nonlinear systems with constraints that are difficult to handle using traditional linear control approaches [12, 32, 29, 35, 54]. Also, MPC's easy-to-understand tuning process gives user the ability to control systems with complex phenomena such as non-minimum phase dynamics, delays and instabilities. One of the most successful applications of MPC today is *Atlas*, which is known as a backflipping robot developed by Boston Dynamics, Inc [33].

More recently, linear and nonlinear representations of aerial vehicle using dynamic mode decomposition with control (DMDc) and sparse identification of nonlinear dynamics (SINDy) have been successfully paired with MPC [23, 34, 31]. The real-time application of the algorithm on cost-effective platforms is left as future work [34]. SINDy may produce models that are more accurate, however they have greater computational complexity. DMDc is less computationally complex and its model is suitable for linear MPC, which is significantly faster than non-linear MPC [31]. This is desirable for practical applications where built-in computational power is limited.

For high-dimensional, complex systems, DMDc provides a lot of benefits [43]. It is founded on the dynamic mode decomposition (DMD) method, a data-driven, *equation-free* architecture that only uses snapshot measurements to reconstruct the system's underlying dynamics [48, 49]. The use of DMD in domains like fluid dynamics, where it has previously been challenging to analyze and build controllers due to the vast number of spatial states necessary for simulation, has seen significant success [14]. DMD has acquired popularity as a method for systems with nonlinear dynamics, due to a strong connection between DMD and Koopman operator theory [26, 46]. DMD can be modified in complex systems with sparse and limited measurements [5]. Sparse measurements have recently been used in a variety of complex systems for control [30]. Due to the limited numbers of sensors; such a situation occurs in many

physical, biological, and engineering systems. Such advantages make DMDc the most applicable alternative as an equation-free control method of complex systems.

1.3 Research Objective

In this thesis, DMDc-MPC framework is designed and presented as a data-driven model discovery and control technique. One focus of the thesis is to discover the unknown dynamics of lateral-directional fighter aircraft dynamics using DMDc algorithm and then using that discovered model for MPC to control the sideslip angle and roll rate of the fighter aircraft. The problem is specially complex since the process is multiple-input and multiple-output (MIMO) system, nonlinear and use noisy sensors. Also, data collection process is tough since the process is open-loop unstable in some regimes.

Another focus of the thesis is to test the designed algorithm in real time, understand the computational cost and how suitable it is for practical applications.

There are six chapters in this thesis. In Chapter 2, the details of data-driven model discovery process and the method used is explained. We describe the theory of model predictive control in Chapter 3. The fighter aircraft model is provided in Chapter 4. The simulation and results are presented in Chapter 5. The thesis is concluded in the final chapter by going over the findings and outlining potential future research.

CHAPTER 2

DATA-DRIVEN MODEL DISCOVERY

Engineering, biology, and physical sciences have a lot to gain from the fast developing discipline of data-driven modeling and control of complex systems [28]. Reliable data from historical records, computer simulations, and experimental data are easily accessible today. Although data is abundant, models are difficult to find. Some of the systems that are the focus of attention today such as robots, DNA structure, aircraft dynamics, stock market or pandemic can be described as a high-dimensional nonlinear systems that occur in time. Although these systems are complex, they can be expressed and modeled in a lower dimensional and understandable way.

DMDc, a robust new method for the identification of dynamical systems from measurement data, will be introduced in this chapter as the data-driven model discovery method used in this thesis. Using the measurements of the input and output data of the high-dimensional nonlinear system, DMDc creates an equation-free linear model of the underlying input and output relationship of the system [43]. DMDc works with instantaneous data (snapshots) and effectively processes measurement data for the analysis of nonlinear dynamical system.

2.1 Dynamic Mode Decomposition with Control

Data driven model discovery method DMDc is introduced and formulated in this section. For the controller design, it is very important to understand the changes in the internal dynamics of the system and how the inputs affect the system. The DMDc algorithm discovers the underlying dynamics of the system and the effect of the inputs on the system separately.

The discovered system model in which the underlying dynamics are revealed and the input and output data collection process are defined in Section 2.1.1. The next section 2.1.2 describes how to manipulate the collected data for system discovery.

2.1.1 Dynamical System with Control

The fundamental assumption that connects the current state x_k and the current control u_k of a linear dynamical system to the future state x_{k+1} can be described as a discrete-time state-space model given by the following:

$$x_{k+1} = \mathbf{A}x_k + \mathbf{B}u_k \tag{2.1}$$

where $x_j \in \mathbb{R}^n$, $u_j \in \mathbb{R}^l$, $\mathbf{A} \in \mathbb{R}^{n\times n}$, and $\mathbf{B} \in \mathbb{R}^{n\times l}$. System states and inputs measured over time at regular intervals are used to create data matrices. The measured system state snapshots $\boldsymbol{X}$ and $\boldsymbol{X}'$ are collected in the following form:

$$\boldsymbol{X} = \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_{m-1} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n\times(m-1)} \tag{2.2}$$

$$\boldsymbol{X}' = \begin{bmatrix} | & | & & | \\ x_2 & x_3 & \cdots & x_m \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n\times(m-1)} \tag{2.3}$$

Control input snapshots collected are in the following matrix:

$$\Upsilon = \begin{bmatrix} | & | & & | \\ u_1 & u_2 & \cdots & u_{m-1} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{l \times (m-1)} \tag{2.4}$$

Hence, (2.1) can be rewritten with the new data matrices:

$$\boldsymbol{X}' = \mathbf{A}\boldsymbol{X} + \mathbf{B}\Upsilon \tag{2.5}$$

By using these three data matrices which are given in (2.2), (2.3) and (2.4), approximations of the linear mappings $\mathbf{A}$ and $\mathbf{B}$ can be found. It is also good to mention that the matrices $\mathbf{A}$ and $\mathbf{B}$ are called the system and control matrix, respectively.

Figure 2.1 illustrates the data collection process of DMDc for the fighter aircraft which is used as a controlled object in this thesis. To explain the process briefly, various pre-defined control surface inputs are given to enable the aircraft to move. The movements of the aircraft are considered as output and measured and recorded with the help of sensors which are accelerometer, gyroscope, and air-data system. In addition, it is assumed that the control surface inputs given to the aircraft are also collected by actuator control electronics (ACE).

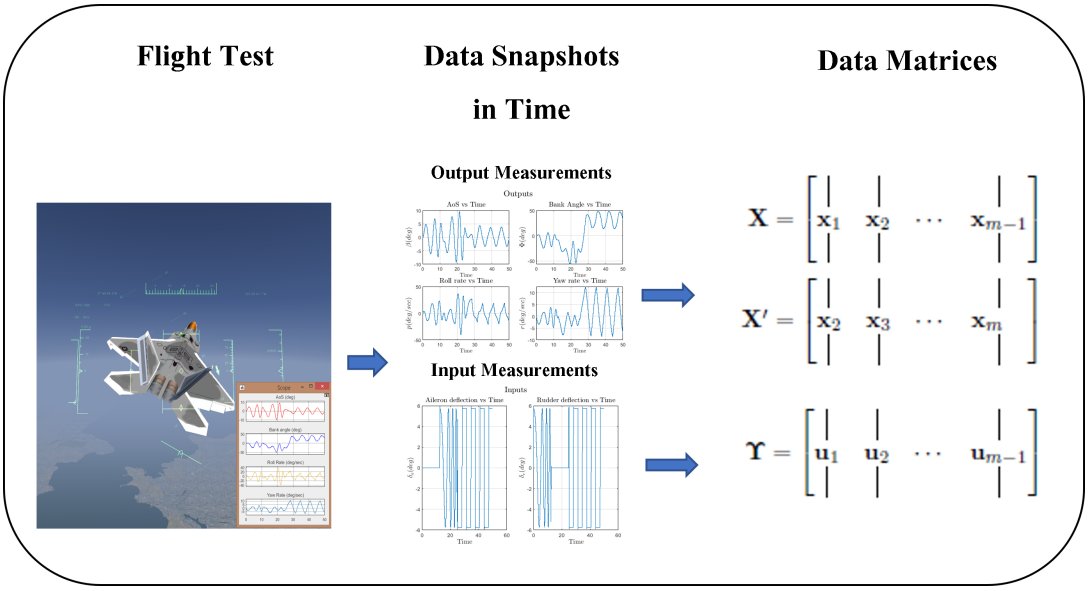


Figure 2.1: Data Collection process of DMDc

In the following Section 2.1.2, how to discover the system matrix $\mathbf{A}$ and control matrix $\mathbf{B}$ from the collected data matrices is described.

2.1.2 Discovery of System and Control Matrices

This section demonstrates that approximations of the matrices $\mathbf{A}$ and $\mathbf{B}$ can both be found from state and control snapshots.

The discrete-time state space system in (2.5) can be manipulated giving the following representation:

$$\boldsymbol{G} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \end{bmatrix} \in \mathbb{R}^{n\times(n+l)} \text{ , } \boldsymbol{\Omega} = \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{\Upsilon} \end{bmatrix} \in \mathbb{R}^{(n+l)\times(m-1)}$$

$$\boldsymbol{X}' = \begin{bmatrix} \mathbf{A} & \mathbf{B} \end{bmatrix} \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{\Upsilon} \end{bmatrix} = \boldsymbol{G}\boldsymbol{\Omega} \tag{2.6}$$

where $\boldsymbol{\Omega}$ contains both the state and control snapshot information. So, we seek a best-fit solution of the operator $\boldsymbol{G}$ which consists of the system matrix A and control matrix B.

The singular value decomposition (SVD) provides a numerically stable matrix decomposition that can be used for a variety of purposes and is guaranteed to exist. We will use the SVD to find pseudo-inverses of non-square matrices [4].

As singular value decomposition exists for every, generally speaking, complex-valued rectangular matrix, the result of applying SVD towards snapshots matrix $\boldsymbol{\Omega}$ defined in (2.6) will give us the following:

$$\boldsymbol{\Omega}_{(n+l)\times(m-1)} = \boldsymbol{U}_{(n+l)\times(n+l)}\boldsymbol{\Sigma}_{(n+l)\times(m-1)}\boldsymbol{V}^{*}_{(m-1)\times(m-1)} \tag{2.7}$$

Here $\boldsymbol{U} \in \mathbb{C}^{(n+l)\times(n+l)}$, and $\boldsymbol{V} \in \mathbb{C}^{(m-1)\times(m-1)}$ are unitary matrices, and $\boldsymbol{\Sigma} \in \mathbb{R}^{(n+l)\times(m-1)}$ is a diagonal matrix with real, nonnegative entries on the diagonal. * denotes the complex conjugate transpose.

When $(n+l) \leq (m-1)$, the matrix $\boldsymbol{\Sigma}$ has at most $(n+l)$ nonzero elements on the diagonal, hence we can write it as:

$$\boldsymbol{\Sigma} = \begin{bmatrix} \hat{\boldsymbol{\Sigma}}_{(n+l)\times(n+l)} & \hat{\mathbf{0}}_{(n+l)\times((m-1)-(n+l))} \end{bmatrix} \tag{2.8}$$

Therefore, it is possible to exactly represent $\boldsymbol{\Omega}$ using economy SVD [4]:

$$\boldsymbol{\Omega} = \boldsymbol{U} \begin{bmatrix} \hat{\boldsymbol{\Sigma}} & \hat{\mathbf{0}} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{V}} & \hat{\boldsymbol{V}}^{\perp} \end{bmatrix}^{*} \tag{2.9}$$

where $\hat{\boldsymbol{V}} \in \mathbb{R}^{(m-1)\times(n+l)}$ and $\hat{\boldsymbol{V}}^{\perp} \in \mathbb{R}^{(m-1)\times((m-1)-(n+l))}$.

The full SVD and economy SVD are shown in Figure 2.2. The columns of $\hat{\boldsymbol{V}}^{\perp}$ span a vector space that is complementary and orthogonal to that spanned by $\hat{\boldsymbol{V}}$. The columns of $\boldsymbol{U}$ are called left-singular vectors of $\boldsymbol{\Omega}$ and the columns of $\boldsymbol{V}$ are right-singular vectors. The diagonal elements of $\hat{\boldsymbol{\Sigma}} \in \mathbb{C}$ are called singular values and they are ordered from largest to smallest. The rank of $\boldsymbol{\Omega}$ is equal to the number of nonzero singular values. So, we may re-write the SVD as follows:

$$\boldsymbol{\Omega} = \boldsymbol{U}\hat{\boldsymbol{\Sigma}}\hat{\boldsymbol{V}}^{*} \tag{2.10}$$

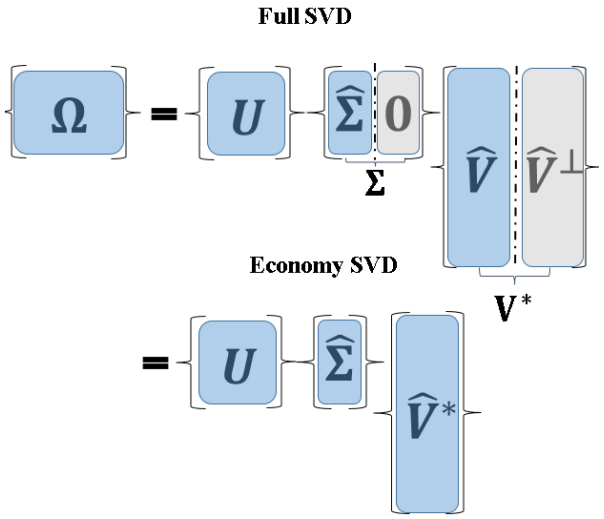


Figure 2.2: Schematic of matrices in the full and economy SVD

Now, by combining (2.10) with the over-constrained equality in (2.6), the least-squared solution $\boldsymbol{G}$ can be found with:

$$\boldsymbol{G} = \boldsymbol{X}'\hat{\boldsymbol{V}}\hat{\boldsymbol{\Sigma}}^{-1}\boldsymbol{U}^{*} \tag{2.11}$$

Since $\boldsymbol{G} = [\mathbf{A}\ \ \mathbf{B}]$, the discrete state-space model approximation is complete.

The procedure above can be written algorithmically as follows [43].

1. **Collection of measured snapshot system states and inputs to construct data matrices:**
   By collecting the measured system states and inputs, $\boldsymbol{X}$, $\boldsymbol{X'}$ and $\boldsymbol{\Upsilon}$ data matrices should be created as defined in (2.2), (2.3) and (2.4). $\boldsymbol{\Omega}$ matrix can be formed after that by combining the $\boldsymbol{X}$ and $\boldsymbol{\Upsilon}$ data matrices into a single matrix.

2. **Calculation of the economy-size SVD of the Ω matrix:**
   The economy-size SVD of the $\boldsymbol{\Omega}$ matrix should be calculated as defined in (2.10) and as a result $\boldsymbol{\Omega} = \boldsymbol{U}\hat{\boldsymbol{\Sigma}}\hat{\boldsymbol{V}}^*$ should be obtained.

3. **Calculation of system matrix A and control matrix B:**
   As defined in (2.11), least-squared solution $\boldsymbol{G}$ can be calculated so that $\boldsymbol{G} = \boldsymbol{X'}\hat{\boldsymbol{V}}\hat{\boldsymbol{\Sigma}}^{-1}\boldsymbol{U}^*$ is obtained. $\boldsymbol{G}$ can be decomposed into system matrix $\mathbf{A}$ and control matrix $\mathbf{B}$ or they can be calculated in the following way:

$$\mathbf{A} = \boldsymbol{X'}\hat{\boldsymbol{V}}\hat{\boldsymbol{\Sigma}}^{-1}\boldsymbol{U_1^*} \tag{2.12}$$

$$\mathbf{B} = \boldsymbol{X'}\hat{\boldsymbol{V}}\hat{\boldsymbol{\Sigma}}^{-1}\boldsymbol{U_2^*} \tag{2.13}$$

   where $\boldsymbol{U_1}$ and $\boldsymbol{U_2}$ are the first n and the last l rows of the left-singular matrix $\boldsymbol{U}$, respectively.

CHAPTER 3

MODEL PREDICTIVE CONTROL

In this chapter, a powerful optimization strategy for feedback control called model predictive control also known as MPC is discussed. This optimal feedback control technique determines the effective control action by using a model to estimate future outputs of a process and solving an optimization problem.

There are some specific reasons to use MPC. First of all, it is capable of handling MIMO systems with interactions between their inputs and outputs. Using conventional controllers like PID, it might be difficult to design MIMO systems because of these interactions [3]. MPC, on the other hand, can concurrently regulate every output while accounting for input-output interactions. Additionally, MPC can manage constraints. Constraints are crucial since violating them may have unfavorable results. With the use of MPC's preview features, reference input changes may be foreseen and the controller's performance can be improved.

Since the 1980s, engineers have employed MPC controllers in process industries. Today, the usage area of MPC has expanded to robotics and aerospace fields thanks to the increasing computing power of processors [54].

3.1 MPC Strategy

The working principle of MPC is summarized in Figure 3.1. To briefly summarize the strategy, the future outputs of the system ($\hat{y}(k+i)$ for $i = 1, \ldots, P$) along a horizon (P) at time k is predicted using the model at hand. By knowing the past inputs and

outputs information of the system up to the moment k, the future response of the system is shaped by optimizing the future inputs $(u(k), u(k+1), ..., u(k+M-1))$ along the control horizon M in order to achieve the desired response. The past inputs of the system $(u(k-i))$ are represented by solid line and the future inputs $(u(k+i)$ for $i = 0, \ldots, M-1)$ are represented by dashed line in Figure 3.1. The control inputs along the control horizon M are optimized to get the desired system outputs. Although the inputs along the control horizon are optimized, only the first element of the optimized inputs can be used at time k, and a new input optimization process starts at the next new time step k.

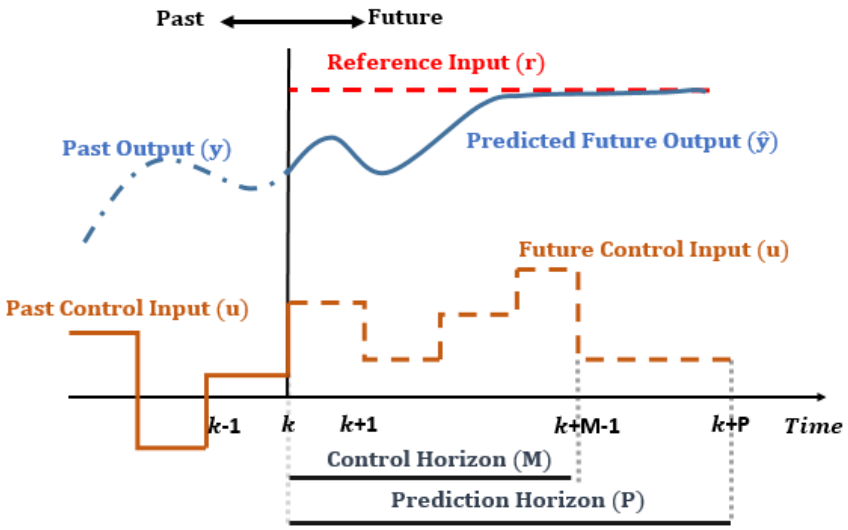


Figure 3.1: MPC Strategy

The basic elements of MPC are shown in Figure 3.2. An error is created by calculating the difference between reference trajectory which is represented by dashed red line in Figure 3.1 and the estimated output. This error is then fed to the optimizer, and future inputs are calculated considering the cost function and constraints. As said before, only the first element of these optimal inputs $u(k)$ is used and the same process continues in the next new time step.

All MPC algorithms have common elements, and different choices for these elements can lead to different algorithms. Basically, the MPC elements are:

- prediction model,
- optimization problem (cost function, constraints and the control law)

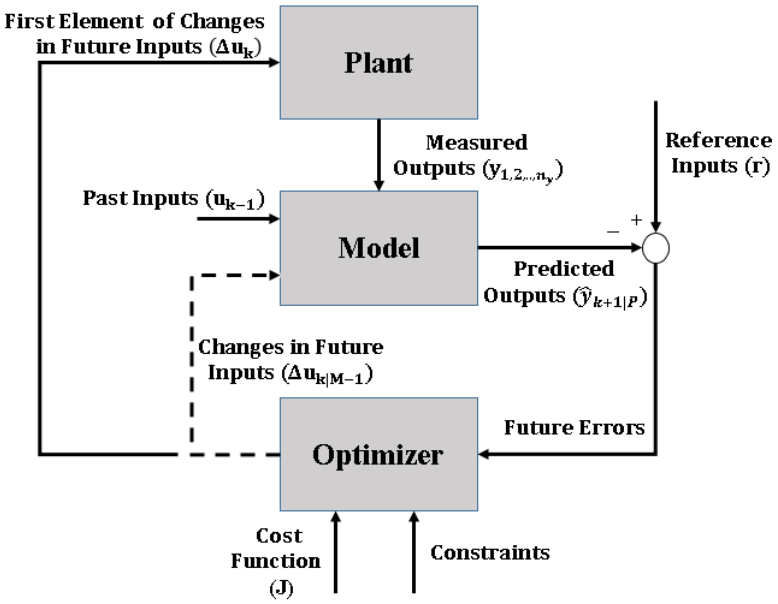


Figure 3.2: Basic Structure of MPC

The most critical and cornerstone element of MPC is the prediction model. In order to achieve a good control performance, all necessary mechanisms should be used to obtain the best possible prediction model. In other words, the prediction model to be created must be capable of fully reflecting the dynamics of the process and allowing the calculation of high precision predictions. It should also be intuitive and allow for theoretical analysis. So in summary, the need to use the prediction model arises from the calculation of the predicted outputs $\hat{y}_{k+1|P}$ and the changes in the future control inputs $\Delta u_{k|M-1}$ vectors at future instants which can be defined as:

$$\hat{y}_{k+1|P} := \begin{bmatrix} \hat{y}_1(t+1) \\ \hat{y}_2(t+1) \\ | \\ \hat{y}_{n_y}(t+1) \\ \vdots \\ \hat{y}_1(t+P) \\ \hat{y}_2(t+P) \\ | \\ \hat{y}_{n_y}(t+P) \end{bmatrix} \in \mathbb{R}^{Pn_y}, \Delta u_{k|M-1} := \begin{bmatrix} \Delta u_1(t) \\ \Delta u_2(t) \\ | \\ \Delta u_{n_u}(t) \\ \vdots \\ \Delta u_1(t+M-1) \\ \Delta u_2(t+M-1) \\ | \\ \Delta u_{n_u}(t+M-1) \end{bmatrix} \in \mathbb{R}^{Mn_u} \tag{3.1}$$

where n_y and n_u are the number of output and input variables, respectively.

3.2 Optimization Problem

It is crucial to make sure that the optimization problem in MPC can be handled in the limited amount of time. The optimization problem is therefore often expressed in one of two basic types:

- Linear programming (LP) problems, where the constraints and objective function are linear functions of the decision variables.
- Quadratic programming (QP) problems, where the objective function is quadratic function of decision variables, whereas the constraints are linear functions of decision variables. In addition, to ensure that there exists a unique optimal solution the QP problem must be convex [19].

LP formulation can be advantageous as it offers the opportunity to find solutions quickly for very high dimensional optimization problems. However, the QP formulation offers the opportunity to make easy to understand choices when adjusting weighting matrix parameters in cost function. [19]. It also results in smoother control actions. Therefore, in the following sections we will concentrate on a QP formulation and go into more detail on how a QP optimization problem can be expressed in MPC.

The QP problem mainly consists of the following elements:

- The cost function to be minimized, which determines the control performance.
- Constraints on the system input variables that the solution must provide.
- Decision that gives the input that minimizes the cost function while satisfying the constraints.

A typical cost function is defined as

$$J_k = (\hat{y}_{k+1|P} - r_{k+1|P})^T \boldsymbol{Q} (\hat{y}_{k+1|P} - r_{k+1|P}) + (\Delta u_{k|M-1})^T \boldsymbol{R} (\Delta u_{k|M-1}). \quad (3.2)$$

The first part of (3.2) is the sum of the square of the difference between the predicted outputs $\hat{y}_{k+1|P}$ and the reference inputs $r_{k+1|P}$ through the prediction horizon. In the

second part, there is the sum of the square of the change of the control inputs $\Delta u_{k|M-1}$ given to the system through the control horizon. The positive semi-definite diagonal weighting matrices $\boldsymbol{Q} \in \mathbb{R}^{(Pn_y)\times(Pn_y)}$ and $\boldsymbol{R} \in \mathbb{R}^{(Mn_u)\times(Mn_u)}$ penalize the output error and control effort.

In general, the quadratic program (QP) for MPC can be expressed in the form:

$$\begin{aligned} \min_{\Delta u_{k|M-1}} \quad & \tfrac{1}{2}\Delta u_{k|M-1}^T \boldsymbol{H} \Delta u_{k|M-1} + f^T \Delta u_{k|M-1}, \\ \text{s.t.} \quad & A\Delta u_{k|M-1} \leq b, \end{aligned}$$

where

- $\Delta u_{k|M-1}$ is the decision variable which is the solution vector;
- $\boldsymbol{H}$ is the positive semi-definite Hessian matrix which is unchanging when the prediction model and weighting matrices are constant while MPC is running;
- A and b are vector of linear constraint coefficients which are constant when the constraints do not change at run time;
- f is a constant vector when the prediction model and output weighting matrix $\boldsymbol{Q}$ do not change at run time.

At the beginning of the control phase, $\boldsymbol{H}$, f, A, and b are calculated once and used for the entire control period. The constraint vector A and vector b are defined in Section 3.2.1 and the derivations of the Hessian matrix $\boldsymbol{H}$ and the vector f in cost function will be presented in Section 3.2.2.

3.2.1 Constraints

Limit values defined for variables during a process control are known as constraints. These constraints are defined by considering the physical limits of the system and it is aimed that the system will work reliably by preventing the formation of absurd inputs that may occur. MPC has become one of the most important control strategies thanks to its ability to handle constraints. The most common types of constraints used for MPC are input, input rates, and output constraints.

3.2.1.1 Input amplitude constraint

The constraints on the magnitude of the input signal can be written as a box-constraint,

$$u_{k|M-1}^{min} \leq u_{k|M-1} \leq u_{k|M-1}^{max}. \tag{3.3}$$

Using the relationship between $u_{k|M-1}$ and $\Delta u_{k|M-1}$,

$$u_{k|M-1} = \Delta u_{k|M-1} + u_{k-1}, \tag{3.4}$$

we convert the box-constraint into two inequality constraints as

$$\begin{aligned} \Delta u_{k|M-1} &\leq u_{k|M-1}^{max} - u_{k-1}, \\ - \quad \Delta u_{k|M-1} &\leq -u_{k|M-1}^{min} + u_{k-1}. \end{aligned} \tag{3.5}$$

3.2.1.2 Input rate constraint

The restrictions on the rate of change of the inputs given to the system are the input rate constraints and defined by

$$\Delta u_{k|M-1}^{min} \leq \Delta u_{k|M-1} \leq \Delta u_{k|M-1}^{max}, \tag{3.6}$$

which is equivalent to

$$\begin{aligned} \Delta u_{k|M-1} &\leq \Delta u_{k|M-1}^{max}, \\ - \quad \Delta u_{k|M-1} &\leq -\Delta u_{k|M-1}^{min}. \end{aligned} \tag{3.7}$$

Inequalities in (3.7) can be written in linear inequality form with the combination of (3.5) in order to get A and b as follows:

$$A = \begin{bmatrix} I \\ -I \\ I \\ -I \end{bmatrix}, \quad b = \begin{bmatrix} u_{k|M-1}^{max} - cu_{k-1} \\ -u_{k|M-1}^{min} + cu_{k-1} \\ \Delta u_{k|M-1}^{max} \\ -\Delta u_{k|M-1}^{min}. \end{bmatrix} \tag{3.8}$$

3.2.2 Decision

A deterministic linear dynamic system can be written as a discrete state space model as

$$x_{k+1} = \mathbf{A}x_k + \mathbf{B}u_k, \tag{2.1}$$

or equivalently,

$$x_{k+1} = \mathbf{A}x_k + \mathbf{B}(u_{k-1} + \Delta u_k). \tag{3.9}$$

Using this state space model, the prediction model can be formulated for instance for $P = 3$, just for showing the iterations: for $k = k + 1$, we write

$$\hat{y}_{k+1} = \mathbf{C}x_{k+1}, \tag{3.10}$$

where $\mathbf{C} \in \mathbb{R}^{(n_y)\times(n_y)}$ is called output matrix and it is selected as identity matrix in this work.
Substituting the equation (3.9) into (3.10), gives

$$\hat{y}_{k+1} = \mathbf{C}(\mathbf{A}x_k + \mathbf{B}(u_{k-1} + \Delta u_k)), \tag{3.11}$$

or equivalently

$$\hat{y}_{k+1} = \mathbf{CA}x_k + \mathbf{CB}u_{k-1} + \mathbf{CB}\Delta u_k. \tag{3.12}$$

For $k = k + 2$ we calculate

$$\begin{aligned} \hat{y}_{k+2} &= \mathbf{C}(\mathbf{A}x_{k+1} + \mathbf{B}u_{k+1}) \\ &= \mathbf{CA}^2x_k + \mathbf{CAB}(u_{k-1} + \Delta u_k) + \mathbf{CB}(u_{k-1} + \Delta u_k + \Delta u_{k+1}) \end{aligned} \tag{3.13}$$

For $k = k + 3$;

$$\begin{aligned} \hat{y}_{k+3} &= \mathbf{C}(\mathbf{A}x_{k+2} + \mathbf{B}u_{k+2}) \\ &= \mathbf{CA}^3x_k + \mathbf{CA}^2\mathbf{B}(u_{k-1} + \Delta u_k) + \mathbf{CAB}(u_{k-1} + \Delta u_k + \Delta u_{k+1}) \\ &\quad + \mathbf{CB}(u_{k-1} + \Delta u_k + \Delta u_{k+1} + \Delta u_{k+2})) \end{aligned} \tag{3.14}$$

By combining equations, (3.12), (3.13) and (3.14), prediction model can be written in matrix form when the prediction and control horizons are equal to P and M as,

$$\underbrace{\begin{bmatrix}\hat{y}_{k+1}\\ \hat{y}_{k+2}\\ \hat{y}_{k+3}\\ \vdots\\ \hat{y}_{k+P}\end{bmatrix}}_{\hat{y}_{k+1|P}} = \underbrace{\begin{bmatrix}\mathbf{CA}\\ \mathbf{CA}^2\\ \mathbf{CA}^3\\ \vdots\\ \mathbf{CA}^P\end{bmatrix}}_{\mathbf{S_x}} x_k + \underbrace{\begin{bmatrix}\mathbf{CB}\\ \mathbf{CB}+\mathbf{CAB}\\ \mathbf{CB}+\mathbf{CAB}+\mathbf{CA}^2\mathbf{B}\\ \vdots\\ \sum_{h=0}^{p-1}\mathbf{CA}^h\mathbf{B}\end{bmatrix}}_{\mathbf{S_{u1}}} u_{k-1}$$

$$+ \underbrace{\begin{bmatrix}\mathbf{CB} & 0 & \cdots & 0\\ \mathbf{CB}+\mathbf{CAB} & \mathbf{CB} & \cdots & 0\\ \mathbf{CB}+\mathbf{CAB}+\mathbf{CA}^2\mathbf{B} & \mathbf{CB}+\mathbf{CAB} & \ddots & 0\\ \vdots & \vdots & \vdots & \vdots\\ \sum_{h=0}^{P-1}\mathbf{CA}^h\mathbf{B} & \sum_{h=0}^{P-2}\mathbf{CA}^h\mathbf{B} & \dots & \sum_{h=0}^{P-M}\mathbf{CA}^h\mathbf{B}\end{bmatrix}}_{\mathbf{S_u}} \underbrace{\begin{bmatrix}\Delta u_k\\ \Delta u_{k+1}\\ \Delta u_{k+2}\\ \vdots\\ \Delta u_{k+M-1}\end{bmatrix}}_{\Delta u_{k|M-1}} \tag{3.15}$$

where $\hat{y}_{k+1|P}$ is trajectory matrix of predicted future variables, $\Delta u_{k|M-1}$ is data matrix of input rate variables, $\mathbf{S_x} \in \mathbb{R}^{(Pn_y)\times n_y}$ is observability matrix for the pair $(\mathbf{C}, \mathbf{A})$, $\mathbf{S_{u1}} \in \mathbb{R}^{(Pn_y)\times n_u}$ is latest input matrix for the pair $(\mathbf{C}, \mathbf{A}, \mathbf{B})$ and $\mathbf{S_u} \in \mathbb{R}^{(Pn_y)\times(Mn_u)}$ is a lower block triangular Toeplitz matrix for $(\mathbf{C}, \mathbf{A}, \mathbf{B})$ matrices. We may re-write (3.15) simply as,

$$\hat{y}_{k+1|P} = \mathbf{S_x} x_k + \mathbf{S_{u1}} u_{k-1} + \mathbf{S_u} \Delta u_{k|M-1} \tag{3.16}$$

and this resulting (3.16) formulation will be the model that MPC will use as a basis for predicting future outputs. Since the output matrix $\mathbf{C}$ is selected as identity in the above equation, it only depends on the current state, x_k. The resulting MPC in this case will be a state feedback type algorithm.

Substituting prediction model, $\hat{y}_{k+1|P}$ of (3.16) into the cost function in (3.2), we obtain the Hessian matrix $\mathbf{H} \in \mathbb{R}^{(Mn_u)\times(Mn_u)}$ and vector $f \in \mathbb{R}^{(Mn_u)}$ as in the following form:

$$\begin{aligned} J_k = \Delta u_{k|M-1}^T \underbrace{(\mathbf{S_u}^T \mathbf{Q} \mathbf{S_u} + \mathbf{R})}_{\mathbf{H}} \Delta u_{k|M-1} \\ + \underbrace{(\mathbf{K_r}^T r_{k+1|P} + \mathbf{K_u}^T u_{k-1} + \mathbf{K_x}^T y_k)^T}_{f} \Delta u_{k|M-1} \end{aligned} \tag{3.17}$$

where

$$\mathbf{K_r} = -\mathbf{QS_u} \tag{3.18}$$

$$\mathbf{K_u} = \mathbf{S_{u1}^T QS_u} \tag{3.19}$$

$$\mathbf{K_x} = \mathbf{S_x^T QS_u} \tag{3.20}$$

Algorithm that is used to calculate Hessian matrix $\mathbf{H}$ and related matrices, $\mathbf{K_r}, \mathbf{K_u}$ and $\mathbf{K_x}$ is given in Appendix A.1. Minimizing the cost function given in (3.17) with respect to $\Delta u_{k|M-1}$, the first order optimality condition

$$\nabla J_k = \overrightarrow{0}, \tag{3.21}$$

yields [40]

$$\Delta u^*_{k|M-1} = -\mathbf{H}^{-1} f \tag{3.22}$$

for the unconstrained optimal solution candidate $\Delta u^*_{k|M-1}$.

To solve the constrained QP problem, active-set solver which uses the QPKWIK algorithm [47] that is based on [13] is selected since it can provide rapid and robust performance for small to medium-size optimization problems.

We note that only the first n_u elements of changes in future control inputs vector $\Delta u^*_{k|M-1}$ is used for control purpose and control input signal u_k is then calculated as $u_k = \Delta u_k + u_{k-1}$.

3.3 Tuning the MPC

The tuning parameters for MPC are the sample time T_s, prediction horizon P, control horizon M, and weighting matrices for predicted errors $\mathbf{Q}$ and control moves $\mathbf{R}$. Below, we investigate these separately.

3.3.1 Sample Time

Qualitatively, as T_s decreases, rejection performance of unknown disturbances typically gets better. The dynamic properties of the plant determine the best T_s value at which performance peaks.

However, the need for computational power rapidly rises as T_s decreases. The best option therefore is a balance between computational effort and performance.

However, the prediction horizon P is an important quantity to consider in MPC. P must vary inversely with T_s if one intends to keep the prediction horizon duration (the product PT_s) constant. As it can be seen from prediction model in the matrix form (3.15), the size of matrices are proportional to P. Therefore, as P increases, the controller memory (RAM) requirements and QP solution time also increase.

Therefore, we consider the following when choosing T_s:

- As a general recommendation, the T_s value should be between 10-15% of the desired minimum closed system response.
- Depending on the improvement of disturbance rejection performance, different T_s values can be used and simulated. T_s value can be reshaped according to the result.
- When the T_s is for more than one second for long duration processes is common in areas such as the chemical industries, the T_s is for less than a second may be required in areas that require rapid response, such as aerospace applications.

3.3.2 Prediction Horizon

At the present time k, the number of future outputs that MPC must evaluate in order to optimize its inputs is known as the prediction horizon, or P.

We have the following observations:

- It is recommended that the P value be determined at the beginning of the MPC design and should not be changed. In other words, the P value should be kept constant while other tuning parameters such as weighting matrices are changed. The prediction horizon P should be chosen large enough so that the MPC can provide internal stability and satisfy constraints.
- If the desired steady state duration is T and sampling time is T_s, the P value should be selected as $T \approx PT_s$.

- Because of the small value selection for P, it may cause undesired plant response by generating unstable controller. If P is already large, followings should be considered:
  - T_s value can be increased.
  - The cost function weights on input rates can be increased.
  - Control horizon can be modified.

3.3.3 Control Horizon

The control inputs time interval to be optimized at each time step k is known as the control horizon, or M. After the control horizon M, the control input moves are kept constant and there is no control movement up to prediction horizon P. The prediction horizon extends the control horizon to predict the final future outputs but without any movement.

The control horizon falls between 1 and the prediction horizon P. Regardless of the optimized control input movements along the control horizon, only the first element of the optimized input sets, that is the input at step k, is used when the MPC starts up and the other values are discarded.

Generally it is recommended to keep $M \ll P$. The reason is the following:

- A small value of M is recommended to reduce the computational cost, as there will be fewer variables to be calculated in the QP resolved in each step.

3.3.4 Output Weighting Matrix

The output weighting matrix $\boldsymbol{Q}$ given in (3.2) is used to penalize the difference between the estimated output and the reference input. The elements in the diagonal of the $\boldsymbol{Q}$ matrix are weighted separately, and the most critical error parameter takes the largest value. The diagonal matrix $\boldsymbol{Q}$ with its diagonal elements $w_{i,j}^{y}$ which are tuning weight for j-th output at i-th prediction horizon step is defined as:

$$\boldsymbol{Q} = diag(w_{1,1}^{y}, w_{1,2}^{y}, \dots w_{1,n_y}^{y}, \dots, w_{P,1}^{y}, w_{P,2}^{y}, \dots, w_{P,n_y}^{y}). \tag{3.23}$$

3.3.5 Input Rate Weighting Matrix

The input rate weighting matrix $\boldsymbol{R}$ given in (3.2) is used to penalize the input rate to be used. The elements in the diagonal of the $\boldsymbol{R}$ matrix are weighted separately and the input which is not desired to be used much is weighted with largest value. The diagonal matrix $\boldsymbol{R}$ with its diagonal elements $w_{i,j}^{\Delta u}$ which are tuning weight for j-th output at i-th prediction horizon step is defined as:

$$\boldsymbol{R} = diag(w_{0,1}^{\Delta u}, w_{0,2}^{\Delta u}, \dots w_{0,n_u}^{\Delta u}, \dots, w_{M-1,1}^{\Delta u}, w_{M-1,2}^{\Delta u}, \dots, w_{M-1,n_u}^{\Delta u}). \tag{3.24}$$

3.4 MPC with Integral Action

When the prediction model used for MPC is not good enough, small steady state errors may occur in response to the given reference input. Therefore, the inclusion of integral action for MPC is an efficient way to eliminate steady state errors that may occur. It also eliminates the unknown slowly changing dynamics and measurement errors [37]. The integral action included in the MPC control strategy to remove the offset is shown in Figure 3.3. The steady state error e that is the difference between the reference input and the measured output is calculated and fed to an integration function which uses Forward Euler method.

For a given step $n > 0$ with time $t(n)$, Forward Euler method calculates the integral output u_{int} as follows:

$$u_{int}(n) = u_{int}(n-1) + K_I[t(n) - t(n-1)]e(n), \tag{3.25}$$

where K_I is the input gain value. When choosing the K_I, the control matrix $\boldsymbol{B}$ discovered by DMDc, should be taken into account. The sign of the K_I will be determined by the sign of the element of the $\boldsymbol{B}$ matrix, which corresponds to the output whose steady state error is to be removed and the input to be used. In addition, the absolute value of K_I should be chosen small considering robustness issues.

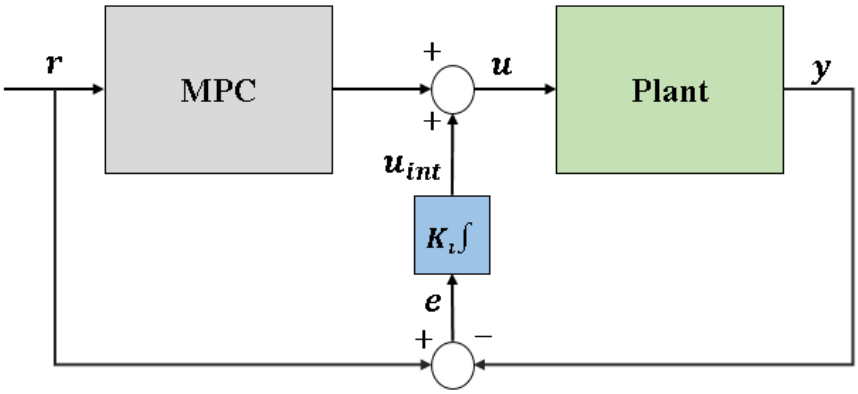


Figure 3.3: MPC with Integral Action

3.5 DMDc and MPC Framework

The proposed DMDc and MPC framework is given in Figure 3.4. The inputs and outputs are collected by giving the predefined inputs to the plant. The collected input and output data are then used for the DMDc model discovery process. Later, the discovered model is transferred to the MPC and used to control the plant. Therefore, after the model discovery, incoming reference commands are switched to commands from the MPC algorithm instead of predefined inputs.

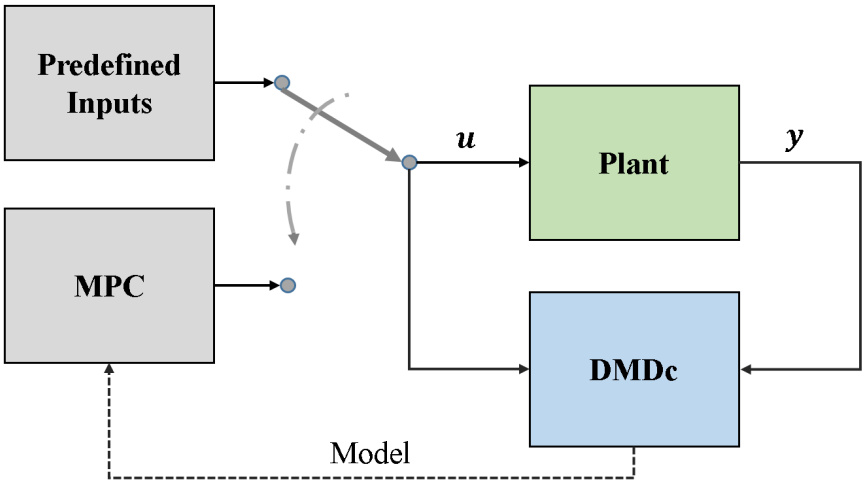


Figure 3.4: Schematic of DMDc and MPC framework

CHAPTER 4

FIGHTER AIRCRAFT MODEL

In this chapter the mathematical model of the fighter aircraft is presented which will be used for model discovery and control. The model represents the hypothetical rudder-aileron controlled fighter aircraft travelling with a speed of 150 m/sec at an altitude of 10,000 m. The model is created in MATLAB's Simulink® environment and, input-output structure is shown in Figure 4.1.

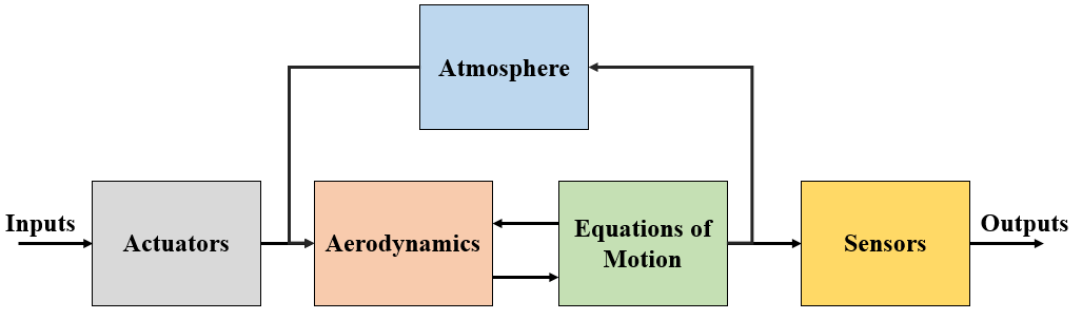


Figure 4.1: Fighter Aircraft Simulation Model

The aircraft model consists of five principal subsystems which are actuators, atmosphere, aerodynamics, equations of motions and sensor models. In the next sections, these subsystem are explained in detail.

4.1 Actuator Models

The actuators are critical component of the flight control system, providing the motive power needed to move the flight control surfaces. Both the aileron and rudder control surfaces must be able to be moved. The deflections of aileron and rudder control

surfaces namely, δ_a and δ_r are shown in Figure 4.2. Since actuators have their own dynamics, they can have a significant impact on the performance of the aircraft and must be considered in the design of a flight control system [41].

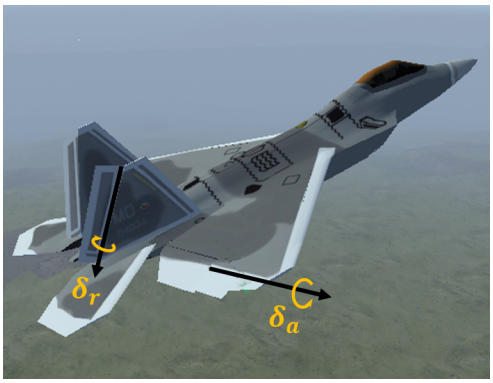


Figure 4.2: Control Surface Deflections

In this section, modelling procedure of actuators for fighter aircraft is presented. The actuator model used by the F-18 aircraft, shared in the literature [6], is used in this work. The actuator mathematical models used for the rudder and aileron surfaces are modeled as a second order transfer function. These transfer functions for aileron and rudder are given as

$$\frac{\delta_a(s)}{\delta_{a_{com}}(s)} = \frac{75^2}{s^2 + 2(0.6)(75)s + 75^2} \tag{4.1}$$

$$\frac{\delta_r(s)}{\delta_{r_{com}}(s)} = \frac{72^2}{s^2 + 2(0.7)(72)s + 72^2} \tag{4.2}$$

In these transfer functions, $\delta_{a_{com}}$ and $\delta_{r_{com}}$ represents the reference commands sent for related actuator to achieve. Finally, the position and rate limits are implemented according to the values given in Table 4.1.

Table 4.1: Actuator model parameters

Control	Position Limit	Rate Limit	Natural Frequency (ω)	Damping (ζ)
$\delta_\mathbf{a}$	$\pm 25°$	$\pm 80°/sec$	75 rad/sec	0.6
$\delta_\mathbf{r}$	$\pm 30°$	$\pm 120°/sec$	72 rad/sec	0.7

4.2 Atmospheric Model

The situation where the forces and moments on an aircraft may vary with the flow-field parameters must be mathematically modeled. It is shown in textbooks on aerodynamics [27] that, for a body of given shape with a given orientation to the free-stream flow, the forces and moments are proportional to the product of free-stream mass density ρ, the square of the free-stream airspeed V_T, and a characteristic area for the body. Therefore, for this purpose, Atmospheric model is designed to calculate air temperature, air density and dynamic pressure according to the International Standard Atmosphere (ISA) model [38]. First of all, the temperature is calculated with respect to altitude change as

$$T = T_0 - Lh, \tag{4.3}$$

where T_0 is the air temperature at sea level, L is the lapse rate, and h is the height from sea level. Air density equation is

$$\rho = \rho_0 \left(\frac{T}{T_0} \right)^{(g/LR)-1} e^{\frac{(h_{tro}-h)g}{RT}}, \tag{4.4}$$

where h_{tro} is the height of the troposphere from sea level, R is the characteristic gas constant for air and ρ_0 is the air density at sea level. Finally, the dynamic pressure, $\bar{q}$ is calculated as

$$\bar{q} = \frac{1}{2}\rho V_T^2. \tag{4.5}$$

4.3 Aerodynamic Model

The forces and moments on the aircraft are calculated using the dimensionless aerodynamic coefficients obtained as a result of wind tunnel test or CFD analysis. Therefore, a suitable model to provide values for the body-axis dimensionless aerodynamic coefficients of the fighter aircraft is required. The body-axis coordinate system is fixed on the aircraft and rotates with the aircraft. The subscript "B" is used to denote this coordinate system. The origin of the body-axis coordinate system is taken to be the center of gravity (c.g.) of the aircraft. The X_B and Y_B axes are taken towards the nose and right side of the aircraft, respectively, while the Z_B axis is taken perpendicular to the X_B and Y_B axes using the right hand rule in body-axis coordinate system. These

axes are illustrated in Figure 4.3. X_B, Y_B and Z_B are the roll, pitch and yaw axes of the aircraft [52]. Aerodynamic coefficient sign conventions are also depicted in Figure 4.3. The roll and yaw aerodynamic moment coefficients, C_l and C_n are defined about X_B and Z_B, respectively. The side force coefficients C_y is defined as positive along the Y_B axis.

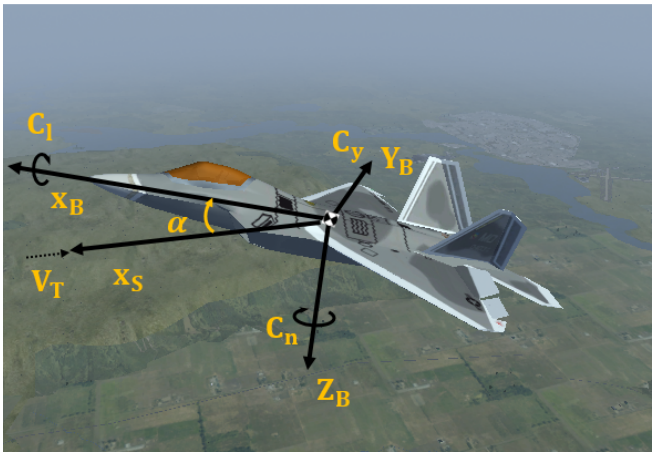


Figure 4.3: Body frame and aerodynamic coefficients

The orientation of the aircraft with respect to the airflow affects the forces and moments that will occur on it. Therefore, the orientation angles which are angle of attack α and the angle of sideslip β are used to determine aerodynamic forces and moments.

The angle of attack α defines the orientation of the stability-axes coordinate system which is denoted by the subscript "S" and it is used for analyzing the effect of perturbations from steady-state flight. As can be seen from the Figure 4.3, it is obtained from the body-axis system by a right-handed rotation, through α, around the body y-axis. The wind-axes system, is denoted by the subscript "W" and is obtained from the stability-axes system by a right-handed rotation, through β, around the z-axis that aligns the wind x-axis, X_W, directly into the relative wind and it can be seen from Figure 4.4.

The equations used to calculate total side force, roll and yaw moment aerodynamic

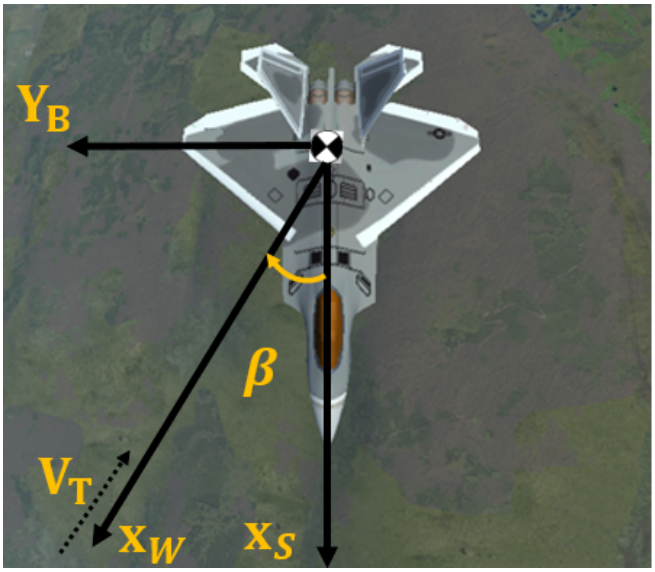


Figure 4.4: Stability frame and related angles

coefficients are

$$C_y = C_{y_{\beta_3}}\beta^3 + C_{y_{\beta_2}}\beta^2 + C_{y_\beta}\beta + C_{y_r} r \frac{b}{2V_T} + C_{y_p} p \frac{b}{2V_T} + C_{y_{\delta a}}\delta_a + C_{y_{\delta r}}\delta_r \quad (4.6)$$

$$C_l = C_{l_{\beta_3}}\beta^3 + C_{l_{\beta_2}}\beta^2 + C_{l_\beta}\beta + C_{l_r} r \frac{b}{2V_T} + C_{l_p} p \frac{b}{2V_T} + C_{l_{\delta a}}\delta_a + C_{l_{\delta r}}\delta_r \quad (4.7)$$

$$C_n = C_{n_{\beta_3}}\beta^3 + C_{n_{\beta_2}}\beta^2 + C_{n_\beta}\beta + C_{n_r} r \frac{b}{2V_T} + C_{n_p} p \frac{b}{2V_T} + C_{n_{\delta a}}\delta_a + C_{n_{\delta r}}\delta_r \quad (4.8)$$

The subscripts of these coefficients indicate the quantity with respect to which the derivative is taken. The quantities p and r are roll and yaw rates and b is the span of the aircraft.

The polynomial coefficients used in the above equations are given in Table 4.2. Some similar aircrafts are reviewed while selecting these coefficients and these are pretty typical values for this kind of aircraft [39, 7]. Note that all coefficients are unitless, thus all angles are in radians.

4.4 Sensor Models

Due to the dynamics of the sensor itself, the filters used in it, calculation delays and analog to digital converters, it can add a phase delay to the controlled system, which must be considered. Therefore, a mathematical modeling of the sensors is required. In

Table 4.2: Aerodynamic Polynomial Coefficients

Side Force	Roll Moment	Yaw Moment
$C_{y_{\beta_3}} = 5.083$	$C_{l_{\beta_3}} = -1.904$	$C_{n_{\beta_3}} = -3.386$
$C_{y_{\beta_2}} = 0.009$	$C_{l_{\beta_2}} = -0.0024$	$C_{n_{\beta_2}} = -0.009$
$C_{y_\beta} = -0.315$	$C_{l_\beta} = -0.0463$	$C_{n_\beta} = 0.112$
$C_{y_p} = 0.1$	$C_{l_p} = -0.25$	$C_{n_p} = -0.035$
$C_{y_r} = 0.4$	$C_{l_r} = 0.065$	$C_{n_r} = -0.3$
$C_{y_{\delta a}} = 0.0213$	$C_{l_{\delta a}} = -0.073$	$C_{n_{\delta a}} = -0.0063$
$C_{y_{\delta r}} = 0.254$	$C_{l_{\delta r}} = 0.02$	$C_{n_{\delta r}} = -0.09$

the literature, there are sensor models that can be used in the initial design stages [53] and these models are used in this work.

The following sensor information are used:

- Body axis angular rates, p and r;
- Angle of sideslip β;
- Body axis roll angle ϕ.

The sensors are modeled as a second order transfer function and they are implemented as given

$$\frac{p_m(s)}{p(s)} = \frac{0.00019s^2 - 0.0173s + 1}{0.000704s^2 + 0.0401s + 1} \tag{4.9}$$

$$\frac{r_m(s)}{r(s)} = \frac{0.00019s^2 - 0.0173s + 1}{0.000704s^2 + 0.0401s + 1} \tag{4.10}$$

$$\frac{\beta_m(s)}{\beta(s)} = \frac{0.116s^2 - 14.437s + 905.92}{s^2 + 29.573s + 908.77} \tag{4.11}$$

$$\frac{\phi_m(s)}{\phi(s)} = \frac{0.3417s^2 - 82.317s + 7161.8}{s^2 + 190.85s + 7162.3} \tag{4.12}$$

The subscript m represents the measured value.

Lastly, it is assumed that deflections, δ_a and δ_r, are measured directly so that $\delta_{a_m} = \delta_a$ and $\delta_{r_m} = \delta_r$.

4.5 Equations of Motion Model

In this work, it is assumed that there are no pitch rate and no cross product of inertia terms on the aircraft. It has constant mass of $30,000\ kg$ (no fuel burning), angle of attack of 20° and total velocity of $150\ m/sec$. Under these assumptions, the lateral-directional nonlinear equations of motion for a rigid aircraft reduce to one force, two moment and one kinematic equations. Firstly, we need to define the aerodynamic force and moment equations to get these force, moment and kinematic equations. F_Y is the aerodynamic body force on the Y_B axis about the center of gravity, M_L and M_N are the rolling and yawing moment on the aircraft in the X_B and Z_B axis and their pictorial representations are given in Figure 4.5. Furthermore, the side force, rolling and yawing moment are defined as:

$$F_Y = \bar{q}SC_y \tag{4.13}$$

$$M_L = \bar{q}SbC_l \tag{4.14}$$

$$M_N = \bar{q}SbC_n \tag{4.15}$$

where S is the wing area of the aircraft respectively. The physical characteristics of the aircraft are given in Table 4.3.

Table 4.3: Fighter Aircraft Physical Characteristics

Symbol	Name	Value
m	Mass	30000 kg
I_{xx}	Rolling Inertia	50000 $kg - m^2$
I_{zz}	Yawing Inertia	450000 $kg - m^2$
S	Wing Area	72 m^2
b	Wing Span	12 m
g	Gravity	9.81 m/sec^2

So, after defining aerodynamic force and moment equations in (4.13)-(4.15), force, moment and kinematic equations that represents the three degrees of freedom (3-

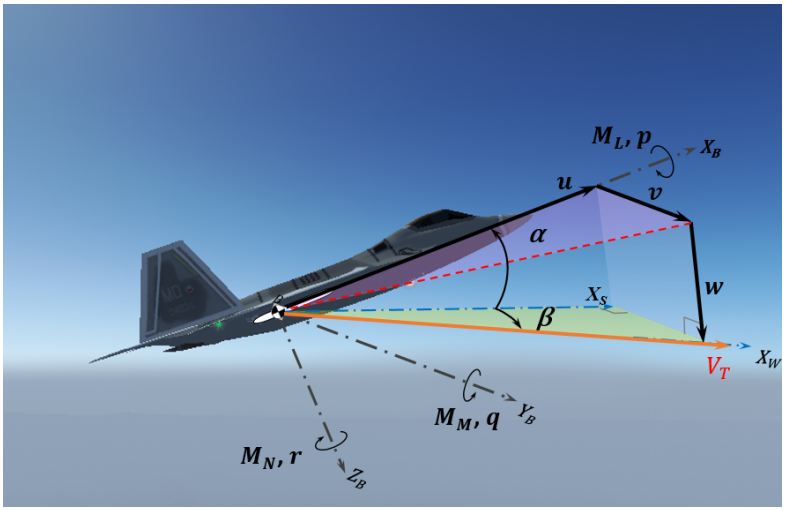


Figure 4.5: Aircraft reference frames

DOF) equations of motion of the aircraft can be defined as in [52, 51, 8]:

$$\dot{v} = \frac{F_Y}{m} - rV_T\cos(\alpha) + pV_T\sin(\alpha) + g\cos(\alpha)\sin(\phi) \tag{4.16}$$

$$\dot{p} = \frac{M_L}{I_{xx}} \tag{4.17}$$

$$\dot{r} = \frac{M_N}{I_{zz}} \tag{4.18}$$

$$\dot{\phi} = p + r\tan(\alpha)\cos(\phi) \tag{4.19}$$

where m is the mass, I_{xx} and I_{zz} are the roll and yaw moment of inertia respectively, V_T is the total velocity (same as free-stream airspeed), v is the body frame velocity in y direction, ϕ is the roll angle, p and r are the roll and yaw rate respectively, g is the gravitational acceleration on the aircraft.

The outputs that are measured in this study are determined as β, p, r and ϕ in Section 4.4. Therefore, conversion from v to β is required. This can be accomplished by using small angle assumption for angle of sideslip β smaller than 15 degrees, we may use [57]

$$\beta = \sin^{-1}\frac{v}{V_T} \approx \frac{v}{V_T} \tag{4.20}$$

and hence, under the assumption of constant total velocity V_T, we have

$$\dot{\beta} = \frac{\dot{v}}{V_T}. \tag{4.21}$$

Finally, the equations (4.17)-(4.19) and (4.21) take the form:

for nonlinear aircraft model,

$$\dot{\beta} = \frac{\bar{q}S}{mV_T}(C_{y_{\beta_3}}\beta^3 + C_{y_{\beta_2}}\beta^2 + C_{y_\beta}\beta + C_{y_r}r\,\frac{b}{2V_T} + C_{y_p}p\,\frac{b}{2V_T} + C_{y_{\delta a}}\delta_a + C_{y_{\delta r}}\delta_r) \\ - r\cos(\alpha) + p\sin(\alpha) + \frac{g\cos(\alpha)\sin(\phi)}{V_T}, \tag{4.22}$$

$$\dot{p} = \frac{\bar{q}Sb}{I_{xx}}(C_{l_{\beta_3}}\beta^3 + C_{l_{\beta_2}}\beta^2 + C_{l_\beta}\beta + C_{l_r}r\,\frac{b}{2V_T} + C_{l_p}p\,\frac{b}{2V_T} + C_{l_{\delta a}}\delta_a + C_{l_{\delta r}}\delta_r), \tag{4.23}$$

$$\dot{r} = \frac{\bar{q}Sb}{I_{zz}}(C_{n_{\beta_3}}\beta^3 + C_{n_{\beta_2}}\beta^2 + C_{n_\beta}\beta + C_{n_r}r\,\frac{b}{2V_T} + C_{n_p}p\,\frac{b}{2V_T} + C_{n_{\delta a}}\delta_a + C_{n_{\delta r}}\delta_r), \tag{4.24}$$

$$\dot{\phi} = p + r\tan(\alpha)\cos(\phi). \tag{4.25}$$

These given equations are nonlinear dynamics that the model discovery algorithm DMDc will try to discover in the next sections. For the studied highly nonlinear aircraft model, all the aerodynamic coefficients in equations (4.22)-(4.24) are non-zero. The coefficients of the nonlinear terms are set to zero for the linear aircraft model. Linear aircraft model for constant α of 20° or $0.349\ rad$, may be considered as

$$\dot{\beta} = \frac{\bar{q}S}{mV_T}(C_{y_\beta}\beta + C_{y_r}r\frac{b}{2V_T} + C_{y_p}p\frac{b}{2V_T} + C_{y_{\delta a}}\delta_a + C_{y_{\delta r}}\delta_r) \\ - r\cos(0.349) + p\sin(0.349) + \frac{g\cos(0.349)\phi}{V_T}, \tag{4.26}$$

$$\dot{p} = \frac{\bar{q}Sb}{I_{xx}}(C_{l_\beta}\beta + C_{l_r}r\,\frac{b}{2V_T} + C_{l_p}p\,\frac{b}{2V_T} + C_{l_{\delta a}}\delta_a + C_{l_{\delta r}}\delta_r), \tag{4.27}$$

$$\dot{r} = \frac{\bar{q}Sb}{I_{zz}}(C_{n_\beta}\beta + C_{n_r}r\,\frac{b}{2V_T} + C_{n_p}p\,\frac{b}{2V_T} + C_{n_{\delta a}}\delta_a + C_{n_{\delta r}}\delta_r), \tag{4.28}$$

$$\dot{\phi} = p + r\tan(0.349). \tag{4.29}$$

True linear state-space model to be used for a comparison in DMDc model discovery process are now written for constant values which are V_T of $150\ m/sec$, ρ of 0.4127 kg/m^3 (density at $10,000m$) and aerodynamic coefficients given in Table 4.2. The

linear state space model describing the body motion become

$$\begin{bmatrix} \dot{\beta} \\ \dot{p} \\ \dot{r} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} -0.023 & 0.342 & -0.938 & 0.064 \\ -3.77 & -0.802 & 0.208 & 0 \\ 0.98 & -0.012 & -0.107 & 0 \\ 0 & 1 & 0.364 & 0 \end{bmatrix} \begin{bmatrix} \beta \\ p \\ r \\ \phi \end{bmatrix} + \begin{bmatrix} 0.001 & 0.019 \\ -5.87 & 1.58 \\ -0.056 & -0.796 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix}. \tag{4.30}$$

Lateral/Directional dynamic stability modes can be determined from the eigenvalues of the system matrix in (4.30). The calculated eigenvalues and corresponding system characteristics are given in Table 4.4 for comparison purposes in DMDc model discovery process.

Table 4.4: Fighter Aircraft Dynamic Modes

Eigenvalue	Damping Ratio (ζ)	Natural Frequency (ω)	Mode
-0.21 $\pm 1.44i$	0.145	1.45	Dutch-roll
-0.517	1	0.517	Roll
0.0063	N/A	N/A	Spiral

4.6 Control Law Model

The control law (CLAW) model is used for comparison purposes only and is an additional model. The control architecture used in this study is similar to that designed for the F/A-18 aircraft [7]. Figure 4.6 shows the architecture of the fighter aircraft flight CLAW.

It should be noted that while determining the gain values, no design method was used, and the selection is made based on the gain values used by past similar aircraft [7]. If enough time is spent, more optimized gains can be selected, resulting in a better-performing CLAW. The main purpose here is to see what can be achieved with the design of similar aircraft without any model dependent design in accordance with the model-free concept.

The objective of the flight CLAW is to provide the necessary stability and handling quality characteristics of the fighter aircraft.

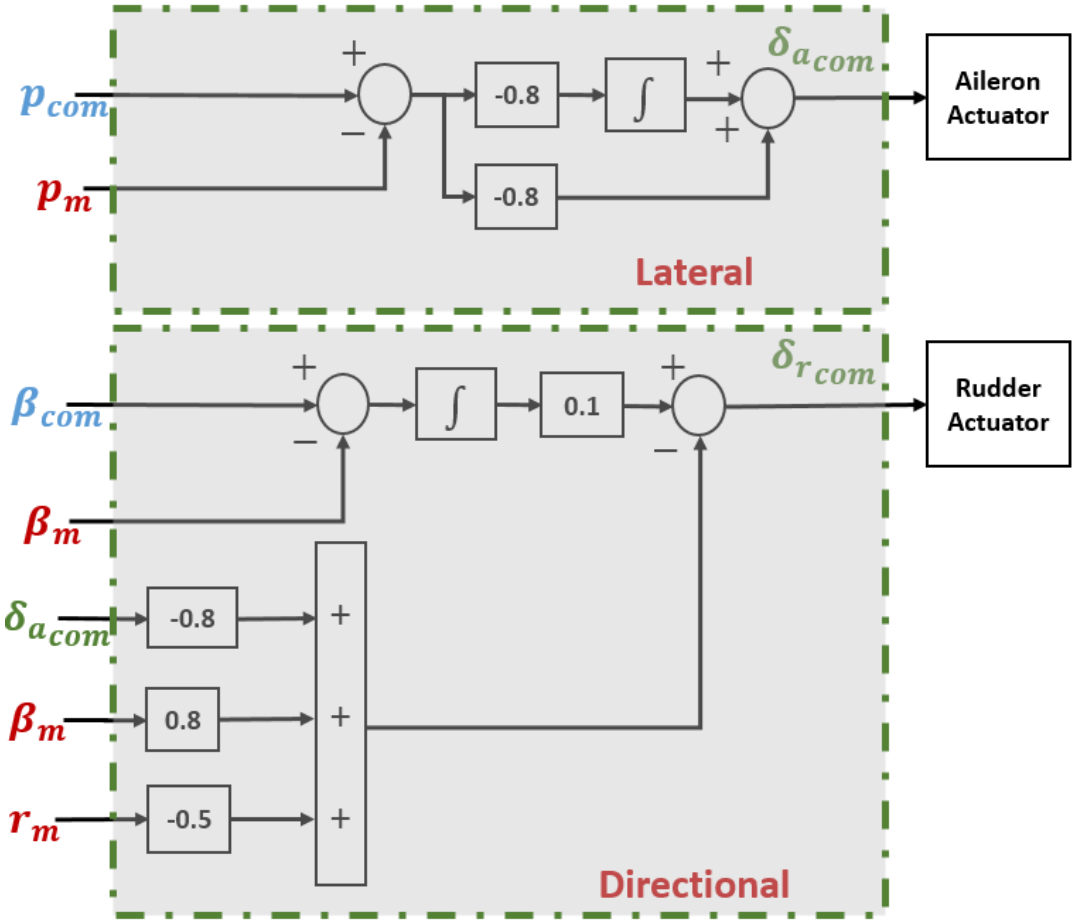


Figure 4.6: Aircraft Flight CLAW

4.6.1 Lateral Control

The roll motion of the aircraft is controlled in the lateral axis. The commanded signal which is denoted as p_{com} is the roll rate command in this channel. To achieve this, the error between the measurement data from the roll gyro sensor and the reference command is multiplied by a gain, improving the aircraft's roll dumping characteristics and handling quality. Since there is high roll damping in aircraft at high speeds, the feedback gain is kept low in this regime, while high feedback gains are used at low speeds. The range of gains used for roll rate feedback is typically 0.8 for low speeds and 0.08 for high speeds [7]. Since the flight condition considered in this study is relatively low speed, -0.8 is used as the proportional feedback gain k_p value to provide necessary roll damping. In addition integrator is added to the roll rate feedback to achieve tighter control of roll rate. The integrator gain $k_{p_{int}}$ is selected as -0.8.

4.6.2 Directional Control

Basically, the Dutch-roll mode characteristic of the aircraft is tried to be improved on the directional axis. For these purposes, the yaw rate r and angle of sideslip β measurement values from the yaw gyro and air data system are fed to the rudder actuator after multiplying with a determined gain. A feedback gain k_r of -0.5 is used to provide yaw damping. The sideslip feedback plays a key role in increasing the lateral stability in the high angle of attack range. Therefore, damping the sideslip motion is critical. Proportional feedback is implemented in this channel. The value of the proportional gain is selected as $k_\beta = 0.8$. The purpose of $\delta_{a_{com}}$ is to provide the component of yaw rate necessary to achieve a stability-axis roll. This cross-connection, known as the aileron-rudder interconnect (ARI). Normally, this ARI gain must be determined as a function of angle of attack and Mach number, however since it is assumed that there are constant angle of attack and Mach number, this gain, k_{ARI} is selected as -0.8 and kept as constant. In addition, integrator is added to the sideslip feedback to achieve tighter control of sideslip which is also used in reported NASA study for X-31 aircraft [20]. Lastly, commanded signal which is denoted as β_{com} is the angle of sideslip command in this channel.

CHAPTER 5

SIMULATION AND RESULTS

The simulation results of the control architecture, namely MPC-DMDc is presented in this section. The programming languages, MATLAB® and Simulink®, are used to carry out these simulations.

The simulation consists of three phases in total. These phases are called training, validation, and control. For the first two phases of the simulation, the aircraft was moved laterally and directionally by giving various control surface inputs. The available dataset is split into two sets, training dataset and validation dataset. The collected training dataset is used for DMDc model discovery process. Afterwards, the discovered model is tested with the validation data which is not used in the identification phase to give information about the precision of the discovered model. Finally, after the training and validation phases are completed, the control phase is started.

It is worth noting that data-driven model discovery process takes place in the air, and only data from the limited flight time is used for training. As a result, the model is not reliant on the past dynamics observed or future dynamics that will be observed. DMDc model discovery algorithm could be used iteratively in the future works to update the model over time, which resulting in Adaptive DMDc-MPC.

The performance of the DMDc depends on the cleanness of the data collected for training, and the noise on the collected data can affect the accuracy of the discovered model. Therefore, Gaussian white noise is added to the sensor measurements and results are analyzed to understand the performance variation with noisy data.

After the training and validation processes completed, the DMDc model is used for

MPC in the control phase, and the control performance is analyzed. Finally, the results are compared with the responses of the traditional CLAW model presented in Section 4.6.

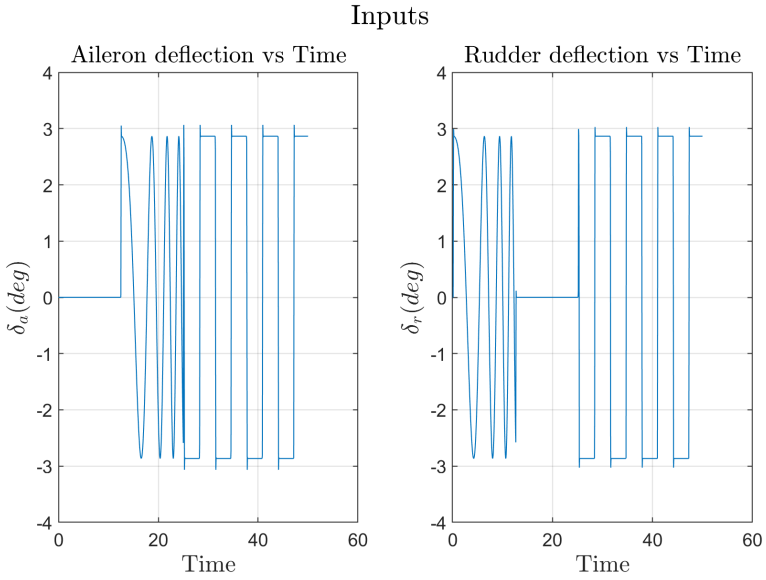
5.1 Training and Validation

The amount of data required to train and validate an accurate model is obtained by running the fighter aircraft model at an $10,000\ m$ altitude with an approximate speed of $150/s$ by giving predefined actuation puts, rudder deflection δ_r, and aileron deflection δ_a. Different actuation inputs are used during the training and validation phases to assess the models' ability to generalize.

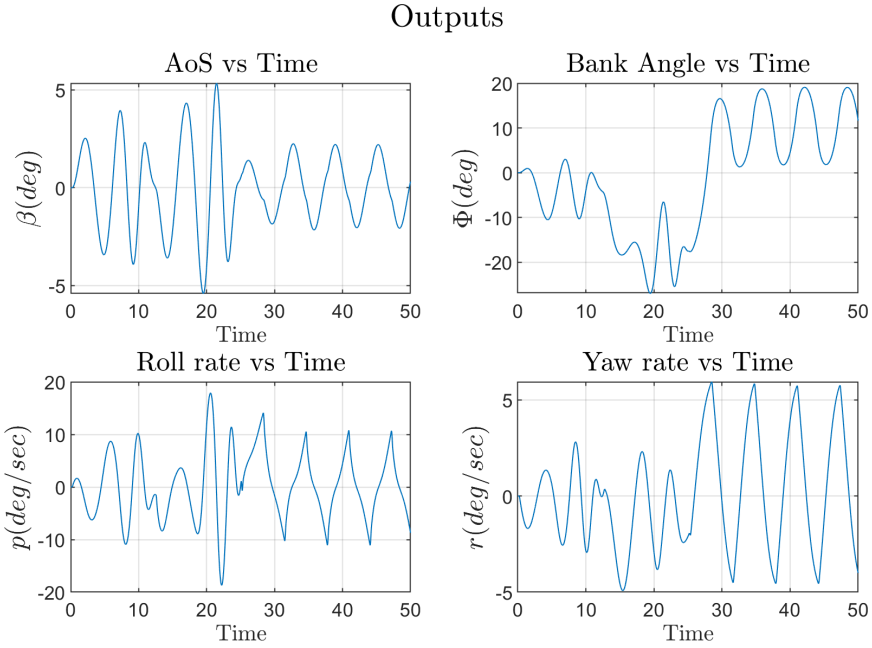
Frequency sweeps for each control deflection inputs are common type of system identification maneuvers [45]. They allow the evaluation of a complete frequency response of the system to the input signal. Usually this type of input signals is applied to one control surface at time. Therefore frequency sweep inputs for one surface at time is used in training phase.

The prediction results of discovered model with the training dataset may be significant, however the final performance of the model should be tested with the validation dataset which is not used in the training phase. Therefore, pulse input are given for δ_r, and δ_a. The inputs used in training and validation phases are shown in Figure 5.1a.

A total of 50 seconds is devoted to the training and validation phases. The input and output data in the first 25 seconds are kept as training dataset, and the data in the next 25 seconds are kept as validation dataset. The output dataset is shown in Figure 5.1b.



(a) Input Dataset



(b) Output Dataset

Figure 5.1: Dataset

The footage from the flight test data collection process visualized using the open software FlightGear Flight Simulator program is shown in the Figure 5.2.

The model configuration parameters used in the simulation as follows;

Model Sample Time	0.001 sec (1000Hz)
Model Solver	Heun

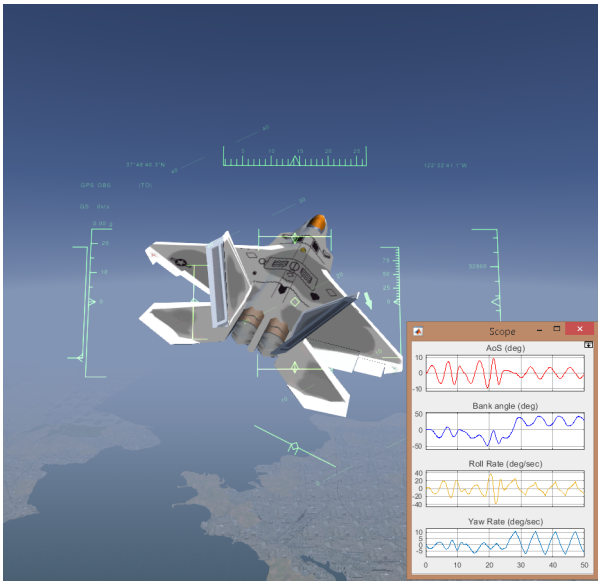


Figure 5.2: Footage from FlightGear

Adding many variables may cause over-fitting, thus it may decrease the accuracy in validation phase which is not the desired case [36]. In order to find the correct sampling rate for data collection process for the DMDc model discovery algorithm, offline analyzes are performed. In these analyzes, the mean of the absolute difference of the eigenvalues of the discovered and true model in clean and noisy measurement conditions is taken into account. These results are given in the Appendix A. The result shows that a data collection rate ΔT_{DMDc} of $0.1\ sec$ or $10\ Hz$ is a suitable choice as given in Figure A.1. This data collection rate is more accessible and practical for sensors.

Since the discovered model will be a discrete linear state-space model, the true linear continuous time state space model in equation (4.30) needs to be discretized so that reasonable comparison can be made. The continuous model is given in the following form:

$$\dot{x} = \mathbf{A}x_k + \mathbf{B}u_k \tag{5.1}$$

Zero-order hold (ZOH) approximation can be made to obtain the true discrete model as such [52]:

$$x_{k+1} = \underbrace{e^{\mathbf{A}T_d}}_{\mathbf{A}_d} x_k + \underbrace{\mathbf{A}^{-1}(e^{\mathbf{A}T_d} - \mathbf{I})\mathbf{B}}_{\mathbf{B}_d} u_k \tag{5.2}$$

where the matrix exponential $e^{\mathbf{A}T}$ is called the *discrete-time transition matrix*, T_d is

the discrete sample time, $\mathbf{A}$ and $\mathbf{B}$ are the continous state-space matrices and $\mathbf{A}_d$ and $\mathbf{B}_d$ are the discretized state-space matrices. The linear discrete state-space model describing the body motion becomes:

$$\begin{bmatrix} \beta_{k+1} \\ p_{k+1} \\ r_{k+1} \\ \phi_{k+1} \end{bmatrix} = \underbrace{\begin{bmatrix} 0.991 & 0.033 & -0.093 & 0.004 \\ -0.360 & 0.917 & 0.037 & -0.001 \\ 0.097 & 0.001 & 0.985 & 0 \\ -0.016 & 0.096 & 0.038 & 1 \end{bmatrix}}_{\mathbf{A}_{d_{True}}} \begin{bmatrix} \beta_k \\ p_k \\ r_k \\ \phi_k \end{bmatrix} + \underbrace{\begin{bmatrix} -0.009 & 0.008 \\ -0.563 & 0.15 \\ -0.005 & -0.079 \\ -0.029 & 0.006 \end{bmatrix}}_{\mathbf{B}_{d_{True}}} \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix} \tag{5.3}$$

where discrete sample time $T_d = 0.1\ sec$, since the sample time of the discovered model $\Delta T_{DMDc} = 0.1\ sec$.

In order to understand how the aircraft behaves during the training and validation phases, linear and nonlinear model responses are compared and shown in the Figure 5.3. It can be said that the true linear model represents actual system quite well in this operational regime. Also it is good to mention that we have additional dynamics which are ignored in our linearization process but they are faster enough so they do not really create significant problem.

In the following sections, the model given in (5.3) will be used as the basis for comparison and the percentage error between the discovered model and the discrete true model will be presented. The time response prediction of the discovered DMDc models which are obtained from clean and noisy measurements are also presented. Finally, the eigenvalues of the true linear continuous time state space model in the Table 4.4 and the eigenvalues of the DMDc model which is converted to the continuous domain are compared.

5.1.1 Clean Data

Using the data from clean measurements for the model discovery process is described in this case. As stated earlier, the discovered model should be tested for the training dataset and the validation dataset which the model has never seen before. Therefore, the results of the true data and the discovered DMDc model are shown for training and validation phases in Figure 5.4. The blue dashed curves show the DMDc results,

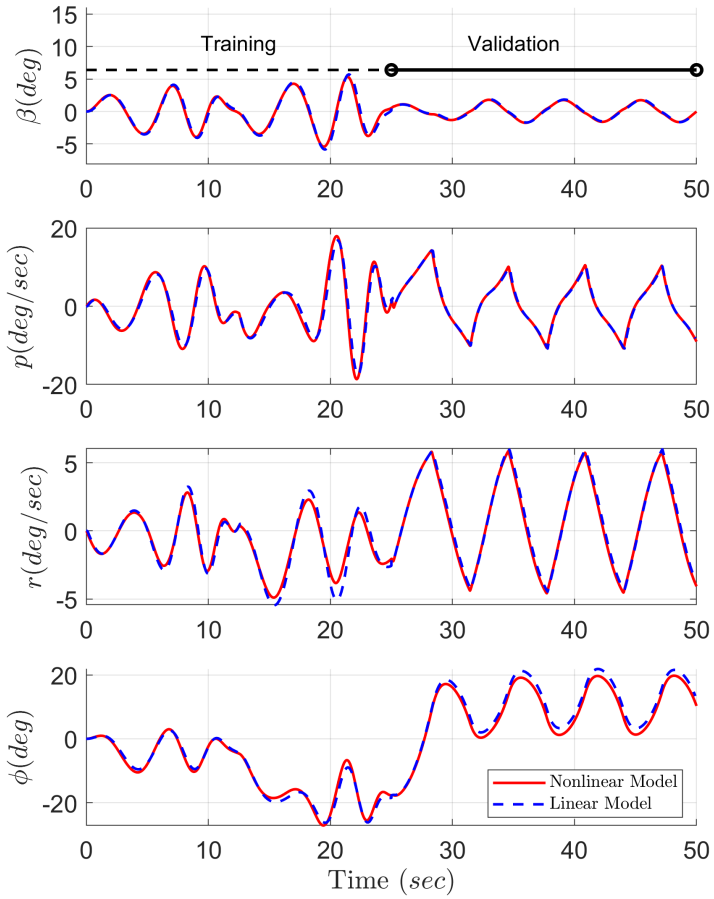


Figure 5.3: Linear vs Nonlinear Model

and the red solid curves the actual airplane responses. The curves for the actual and predicted responses in the Figure 5.4 are almost coincident. It can be interpreted that the parameters of the discovered model are quite close to the actual model parameters.

The discrete linear model that DMDc discovered is given as

$$\begin{bmatrix} \beta_{k+1} \\ p_{k+1} \\ r_{k+1} \\ \phi_{k+1} \end{bmatrix} = \underbrace{\begin{bmatrix} 0.985 & 0.033 & -0.092 & 0.006 \\ -0.421 & 0.915 & 0.034 & -0.001 \\ 0.085 & -0.001 & 0.985 & 0 \\ -0.027 & 0.094 & 0.037 & 1 \end{bmatrix}}_{\mathbf{A_{d_{DMDc}}}} \begin{bmatrix} \beta_k \\ p_k \\ r_k \\ \phi_k \end{bmatrix} + \underbrace{\begin{bmatrix} -0.011 & 0.009 \\ -0.576 & 0.159 \\ -0.009 & -0.077 \\ -0.041 & 0.009 \end{bmatrix}}_{\mathbf{B_{d_{DMDc}}}} \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix} \tag{5.4}$$

In order to understand the closeness of the discovered model to the real model, the

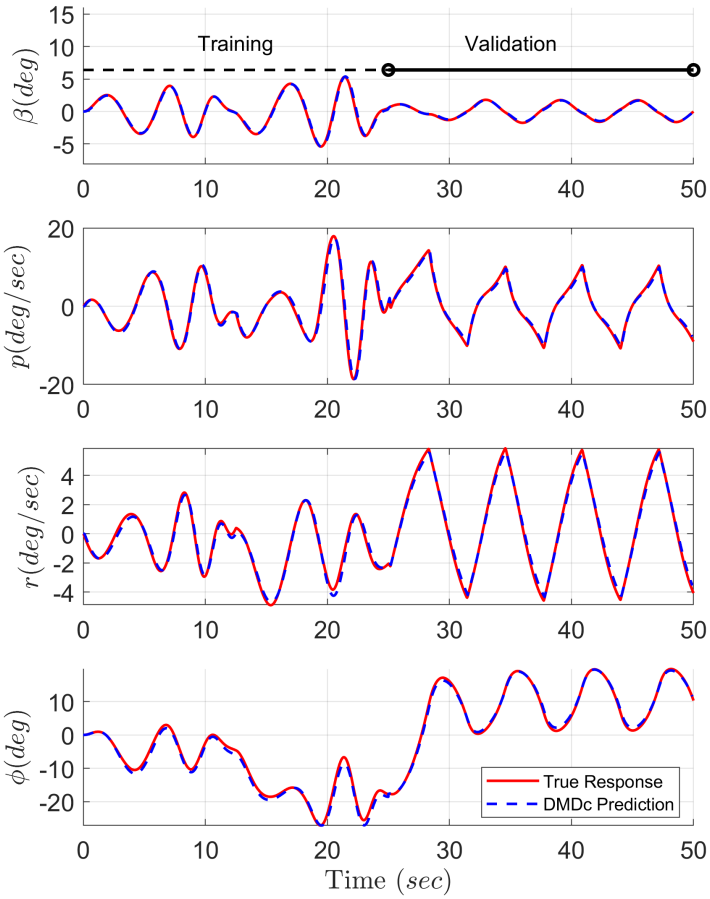


Figure 5.4: Training and validation results from clean data

error difference between the two models is examined and the error percentage is calculated for each element of the matrices $\mathbf{A_d}$ and $\mathbf{B_d}$. So, assuming that $X_d \in \mathbb{R}^n$ is a vector that contains all the individual elements of matrices $\mathbf{A_d}$ and $\mathbf{B_d}$ separately, starting with the rows, the following formulas are used to calculate the model error.

$$E(i) = \frac{|X_{d_{DMDc}}(i) - X_{d_{true}}(i)|}{|X_{d_{DMDc}}(i)|} 100\% \tag{5.5}$$

the mean of the error of the matrix elements is calculated as

$$ME = \frac{E(i)}{n} \tag{5.6}$$

The error results for each elements of matrices $\mathbf{A_d}$ and $\mathbf{B_d}$ are shown as a bar graph and the mean of the errors is represented by a red line in Figure 5.5. The mean errors are calculated as 25% and 17% for matrices $\mathbf{A_{d_{DMDc}}}$ and $\mathbf{B_{d_{DMDc}}}$, respectively.

Although it makes sense to look at the percentage error of each element to find out which element in the matrix is causing the problem, some of these elements in

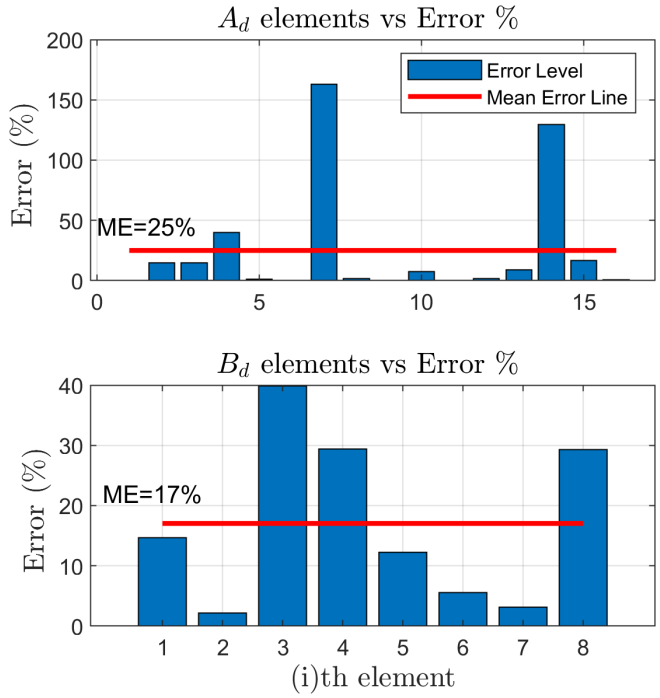


Figure 5.5: Error percentage for each matrix element in the clean data case

$\mathbf{A_{d_{DMDc}}}$ and $\mathbf{B_{d_{DMDc}}}$ may be really insignificant and they may show a very large percentage error even though it does not really mean much. The contribution of a single element in the matrix to represent the dynamic may be very small indeed, and so the large percentage error may not really mean much. Therefore, eigenvalues comparison is a much better way of telling whether the identified DMDc model is close enough to the true model or not. The eigenvalues of the DMDc model which is converted into the continuous domain are given in the Figure 5.6 along with that of the true linear model. The error percentages between true and DMDc model are also given in Table 5.1. While small error values 2% and 1% are obtained for Dutch roll and Roll modes, respectively, 28% error value is calculated for Spiral mode and unstable root is found for this mode as in the true linear model. Basically, Spiral mode develops slowly and it effects long-term bank angle response. Since the predictions for MPC will be made for a short period of time, the eigenvalue error values in Dutch-roll and Roll modes, which are rapidly developing and affect the transient response, are of greater importance.

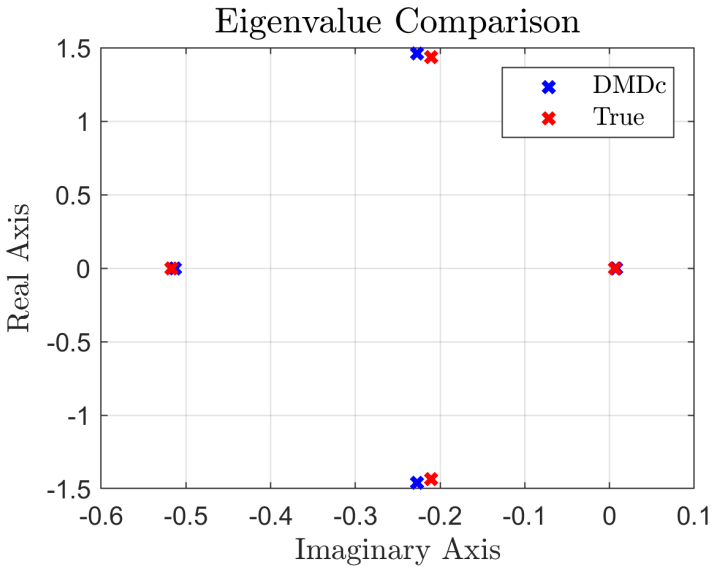


Figure 5.6: Eigenvalue comparison for clean data case

Table 5.1: Eigenvalue comparison in the clean data case

True Linear Model	DMDc	Percent error	Mode
-0.21+1.436i	-0.22+1.46i	2.08%	Dutch-roll
-0.21-1.436i	-0.22-1.46i	2.08%	Dutch-roll
-0.517	-0.513	0.82%	Roll
0.0063	0.008	28.22%	Spiral

So in general, because of small eigenvalue error and the time response of the discovered DMDc model is very close to the real response in the training and validation phases, these can be interpreted as the DMDc performing well for clean measurement cases.

5.1.2 Noisy Data

A clean measurement data may not be obtained by the sensors due to structural or environmental vibrations on the aircraft. Although the sensors' noise can be filtered out, just to understand how noisy measurements affect the model discovery process, different noise levels are added to the clean sensor measurements and their effects are examined in this section. Noise level is calculated as the ratio of root mean square

(RMS) of the noise to that of the clean data and given by

$$\text{Noise Level}(\%) = \frac{\text{RMS}(x_{noisy} - x_{clean})}{\text{RMS}(x_{clean})} 100\% \tag{5.7}$$

Two different noise levels namely, medium and high, are defined in Table 5.2.

Table 5.2: Noise Levels

	Noise Level (%)
Medium	7
High	15

The measurement noise is generated by Gaussian white noise $\mathcal{N}(0,\, \sigma)$ and added to the clean measurements. Noises are added for angular rates p, r and angle of side-slip β signals and it is assumed that the roll angle ϕ signal is relatively noise free [53]. The standard deviations corresponding to each noise level are given in Table 5.3

Table 5.3: Standard Deviations

	β	p	r
Medium	0.0025	0.01	0.0032
High	0.0056	0.02	0.071

In order to understand how the added noise affects on measurements, the clean, medium and high noise sensor measurement results during the first 50 seconds of training and validation are depicted in Figure 5.7.

5.1.2.1 Medium Noisy Data

The scenario in which moderate noise sensor measurements are used for the model discovery process is explained in this section. As it is done in the previous section, the actual and predicted results are compared to understand the model accuracy in the training and validation phases. These results are given in Figure 5.8.

Although the curves do not overlap in Figure 5.8 as it is in clean case, it cannot be said that the discovered model makes a bad prediction. While the difference is quite small for β and p, the difference is widening for r and ϕ.

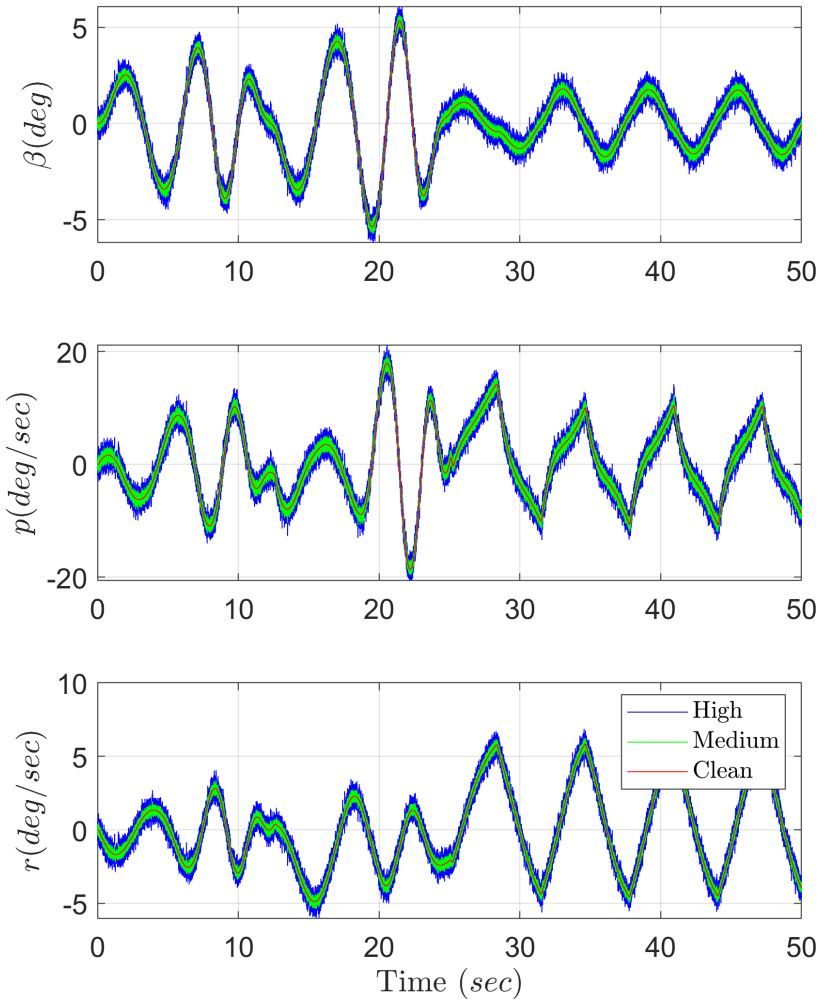


Figure 5.7: Noise Levels

The discrete linear model that the DMDc discovered is given by

$$\begin{bmatrix} \beta_{k+1} \\ p_{k+1} \\ r_{k+1} \\ \phi_{k+1} \end{bmatrix} = \underbrace{\begin{bmatrix} 0.977 & 0.027 & -0.111 & 0.008 \\ -0.450 & 0.896 & -0.025 & 0.007 \\ 0.075 & -0.008 & 0.961 & 0.003 \\ -0.029 & 0.093 & 0.029 & 1 \end{bmatrix}}_{\mathbf{A_{d_{DMDc}}}} \begin{bmatrix} \beta_k \\ p_k \\ r_k \\ \phi_k \end{bmatrix} + \underbrace{\begin{bmatrix} -0.018 & 0.007 \\ -0.600 & 0.152 \\ -0.018 & -0.079 \\ -0.042 & 0.008 \end{bmatrix}}_{\mathbf{B_{d_{DMDc}}}} \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix} \tag{5.8}$$

The error results for each elements of matrices are shown as a bar graph and the mean of the errors is represented by a red line in Figure 5.9. By looking at the mean error values, it can be said that the error is increased by 50% compared to the clean case.

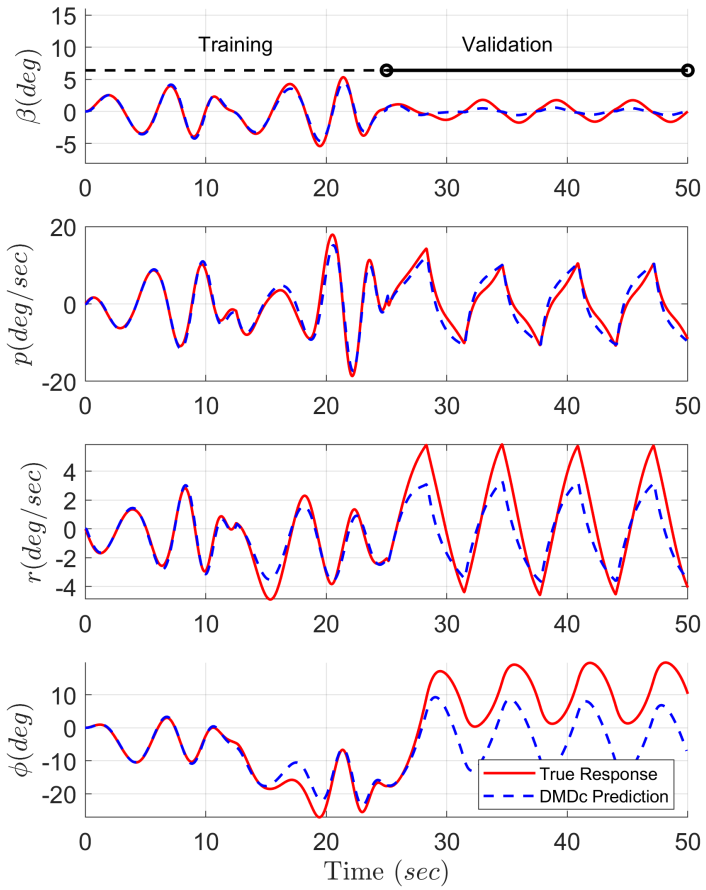


Figure 5.8: Training and validation results for medium noise

As in the clean case, the eigenvalues of the DMDc model which is converted into the continuous domain are given in the Figure 5.10 along with that of the true linear model. The error percentages between true and DMDc model are also given in Table 5.4. Although the error value for the Dutch-roll mode is decreased to 1.26%, the values are increased for Roll and Spiral modes comparing to clean measurement case.

Table 5.4: Eigenvalue comparison in the medium noise case

True Linear Model	DMDc	Percent error	Mode
-0.21+1.436i	-0.21+1.45i	1.26%	Dutch-roll
-0.21-1.436i	-0.21-1.45i	1.26%	Dutch-roll
-0.517	-1.128	118.04%	Roll
0.0063	0.032	408.52%	Spiral

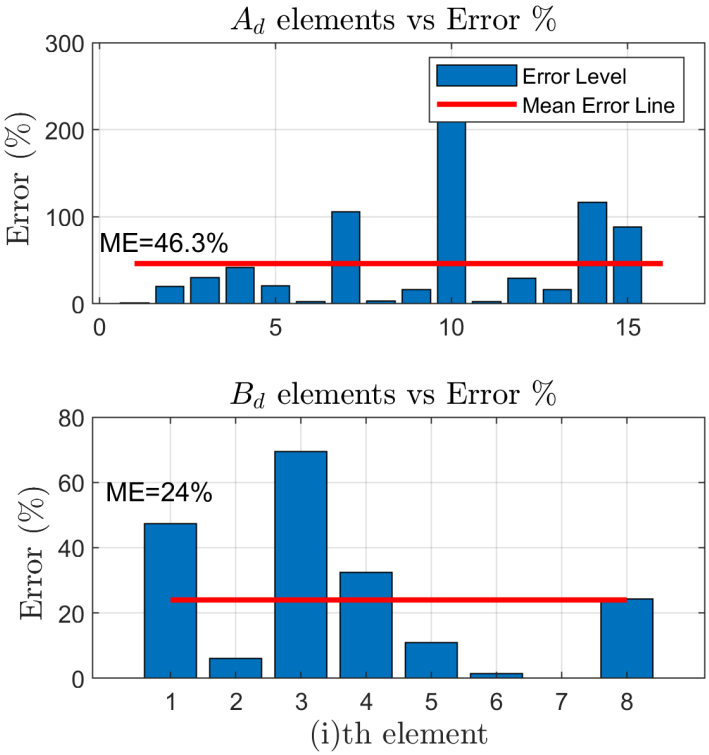


Figure 5.9: Error percentage for each matrix element for medium noise case

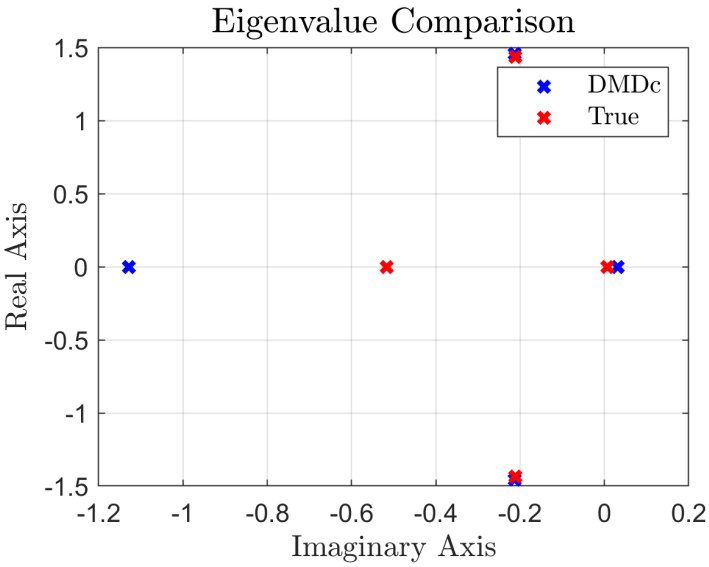


Figure 5.10: Eigenvalue comparison for medium noise case

5.1.2.2 High Noisy Data

Finally, the model discovery results based on high noise measurements are shared in this section. The actual and predicted results are shown in Figure 5.11 for the training and validation phases. It can be observed that the difference between the curves is started to widen considerably. However, it is also observed that the predictions made

by the discovered model can still follow the real trend.

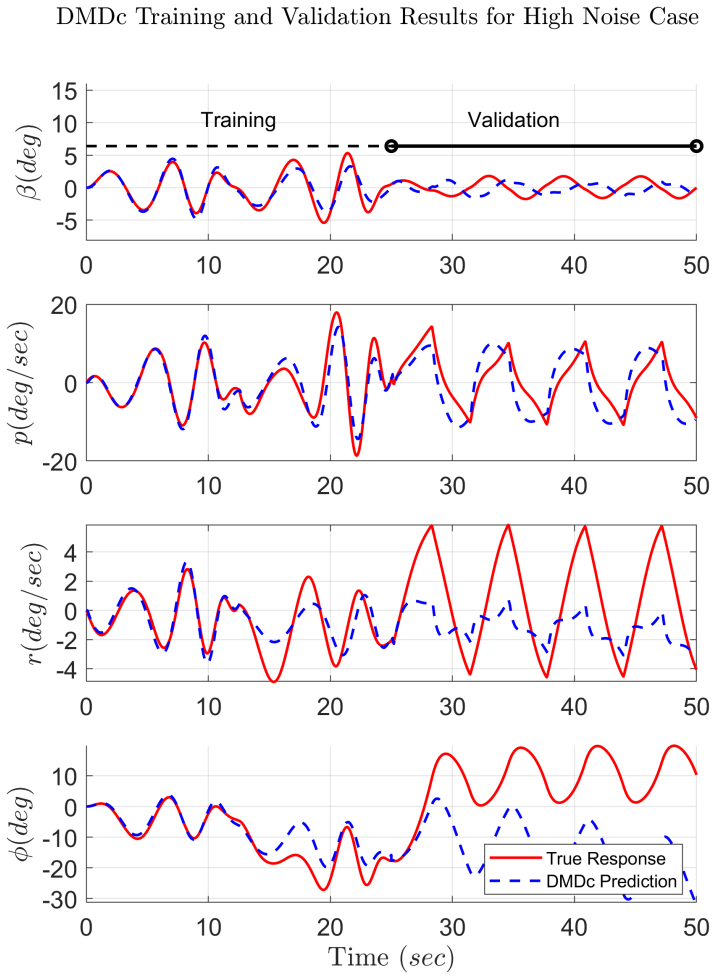


Figure 5.11: Training and validation results for high noise

The discovered discrete linear model that DMDc generated is given by

$$\begin{bmatrix} \beta_{k+1} \\ p_{k+1} \\ r_{k+1} \\ \phi_{k+1} \end{bmatrix} = \underbrace{\begin{bmatrix} 0.951 & 0.011 & -0.167 & 0.014 \\ -0.525 & 0.850 & -0.178 & 0.026 \\ 0.040 & -0.029 & 0.885 & 0.011 \\ -0.036 & 0.088 & 0.008 & 1.001 \end{bmatrix}}_{\mathbf{A_{d_{DMDc}}}} \begin{bmatrix} \beta_k \\ p_k \\ r_k \\ \phi_k \end{bmatrix} + \underbrace{\begin{bmatrix} -0.041 & 0.002 \\ -0.665 & 0.138 \\ -0.049 & -0.086 \\ -0.050 & 0.007 \end{bmatrix}}_{\mathbf{B_{d_{DMDc}}}} \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix} \tag{5.9}$$

The error results for each elements of matrices are shown as a bar graph and the mean of the errors is represented by a red line in Figure 5.12. Looking at the results, it can be concluded that the value of mean error is doubled compared to the medium noise level case.

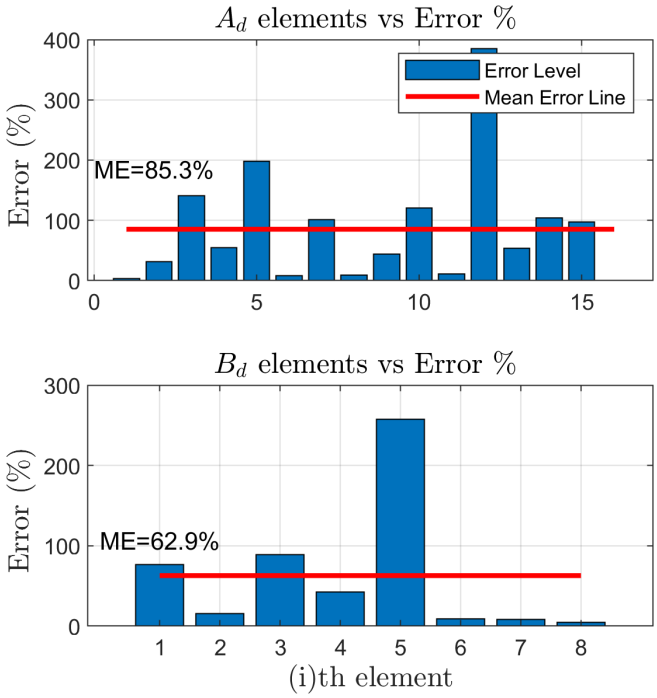


Figure 5.12: Error percentage for each matrix element for high noise case

The eigenvalues of the DMDc model are given in the Figure 5.13 along with that of the true linear model. The error percentages between true and DMDc model are also given in Table 5.5. Although the error value for the Dutch-roll mode is still low as 2.57%, the values are increased for Roll and Spiral modes comparing to medium noise case. It can be concluded that DMDc model discovery process is significantly affected by high noise.

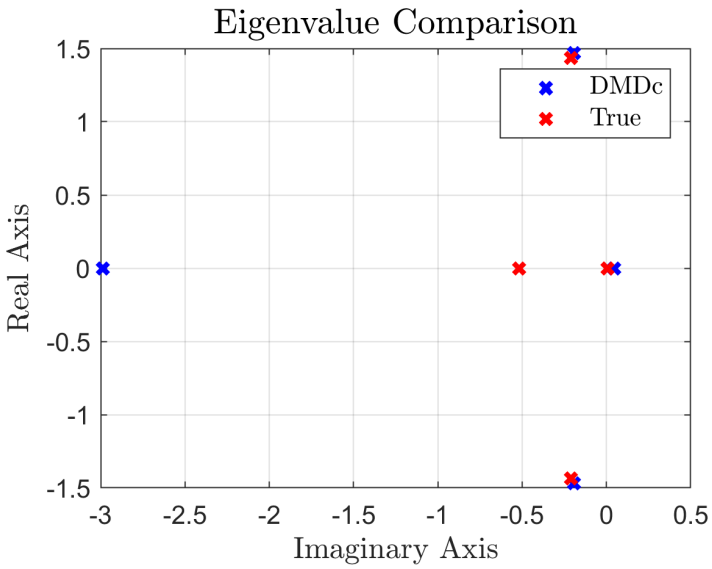


Figure 5.13: Eigenvalue comparison for high noise case

Table 5.5: Eigenvalue comparison in the high noise case

True Linear Model	DMDc	Percent error	Mode
-0.21+1.436i	-0.19+1.47i	2.57%	Dutch-roll
-0.21-1.436i	-0.19-1.47i	2.57%	Dutch-roll
-0.517	-2.98	477.62%	Roll
0.0063	0.046	634.45%	Spiral

5.2 Control Performance Results

After 50 seconds of training and validation phases, 20 seconds of control phase begins. Normally, the reference control inputs given to the aircraft are given by the pilot with the help of the stick and pedal in the cockpit. The lateral stick input commands the reference roll command, and the pedal input controls the sideslip angle. In the control phase, these reference inputs are predefined and given. The responses of the MPC and the classical controller are presented in this section and some evaluation criteria are defined to understand how good the responses are for each controller.

5.2.1 Evaluation criteria

The designed controller should meet some requirements. For the purposes of verifying this, this subsection sets out a set of specific evaluation criteria against which the design should be measured. These requirements are based on MIL-Specification [56] and GARTEUR [53] documents. The criteria are divided into two subclasses which are:

- Robustness,
- Performance.

5.2.1.1 Robustness requirements

The closed loop system should be able to withstand the application of independent gain and delay offsets at the input of each one of the actuators as shown in Figure 5.14

without becoming unstable. The gain margin (GM) and delay margin (DM) that the system must withstand are defined in terms of dB and seconds. This analysis is carried out for only noise free scenario.

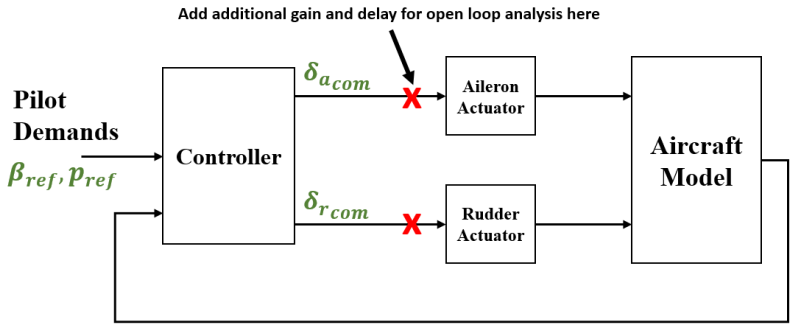


Figure 5.14: Closed loop system showing point for analysis

5.2.1.2 Performance requirements

There are some performance requirements that the aircraft must meet while performing a roll maneuver. Followings are the requirements:

- The maximum roll rate p_{max} should be at least $30°/sec$.
- The roll time constant τ_r should not exceed $1\ sec$. How the τ_r value should be calculated is explained in the Figure 5.15.
- The coupling in sideslip due to roll should be minimised and not exceed $\pm 2°$.

All robustness and performance requirements that the aircraft must meet during the control phase are summarized in the Table 5.6.

Table 5.6: Lateral-Directional Design Criteria

Requirement Name	Requirement
Single Loop GM $[dB]$	GM ≥ 6
Single Loop DM $[sec]$	DM ≥ 0.1
Effective Roll Mode Time Constant $[sec]$	$\tau_r \leq 1$
Maximum Roll Rate $[°/sec]$	$\lvert p_{max} \rvert \geq 30$
Sideslip due to Roll $[°]$	$\lvert \beta \rvert \leq 2$

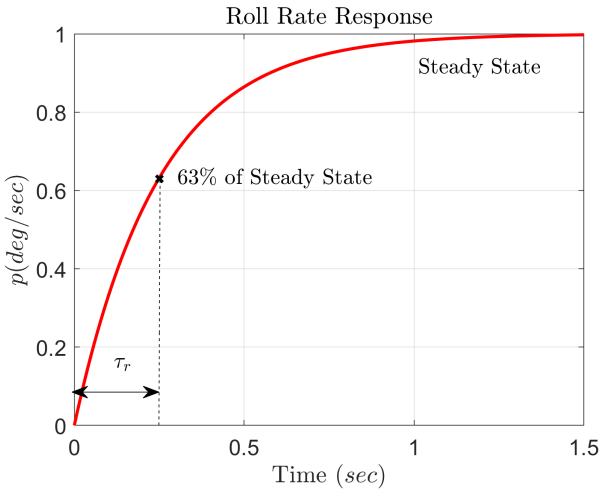


Figure 5.15: Roll mode time constant calculation

5.2.1.3 Actuator usage comparison

The frequent use of the actuator is not desirable in terms of both practical and energy consumption. In order to understand the usage of the actuator, some metrics are determined and these are defined as Control Expenditure (CE) and Control Rate Expenditure (CRE)

$$\text{CE} = \int_{T_{50}}^{T_{70}} |\delta| \tag{5.10}$$

$$\text{CRE} = \int_{T_{50}}^{T_{70}} |\dot{\delta}| \tag{5.11}$$

The T_{50} and T_{70} represents the control phase time interval. There are no specific values defined for these metrics, CE and CRE. These values will be used for comparison purposes only for the MPC and Classic Controller.

In order to test these criteria in Table 5.6, the $30°/sec$ roll rate command p_{com} and the $0°$ sideslip command β_{com} are given to the controllers as reference inputs and the MPC parameters are chosen accordingly. While weighting the outputs for MPC, only the outputs of sideslip and roll rate are penalized in the $\boldsymbol{Q}$ matrix. There is no penalty for yaw rate and bank angle, so that the diagonal elements of the $\boldsymbol{Q}$ matrix which is represented as $w^y_{i,(1-4)}$, is chosen as $[10,\ 1,\ 0,\ 0]$ at each i-th predicton horizon step by considering the units and possible magnitudes of the outputs.

On the other hand, the elements in the diagonal of the $\boldsymbol{R}$ matrix is represented as

$w^{\Delta u}_{i,(1-2)}$ and the weights of inputs rate, namely the rudder and aileron input rate, $\dot{\delta}_r$ and $\dot{\delta}_a$ are chosen as equal: $[0.2,\ 0.2]$ at each i-th predicton horizon step.

The sampling time T_s of the MPC is taken to be the same as the sampling time of the discovered model, ΔT_{DMDc}. The input and input rate constraints are chosen by considering the actuator position and rate limits which are given in Table 4.1.

Finally, the integral action is used to eliminate steady state errors in β and p. δ_r input is used to remove the error in β and δ_a input is used to remove the error in p. Therefore, when choosing the sign of the integral input gains, the elements corresponding to $\beta - \delta_r$ and $p - \delta_a$ in the $\mathbf{B_{d_{DMDc}}}$ matrix are taken into account.

All the design parameters used for MPC are given in the Table 5.7.

Table 5.7: MPC Design Parameters

MPC Design Parameters	Values
Sample Time (T_s)	0.1
Prediction Horizon (P)	40
Control Horizon (M)	10
Output Weight ($w^{y}_{i,(1-4)}$)	$[10,\ 1,\ 0,\ 0]$
Input Rate Weight ($w^{\Delta u}_{i,(1-2)}$)	$[0.2,\ 0.2]$
Input Constraints	$[\pm 25^\circ,\ \pm 30^\circ]$
Input Rate Constraints	$[\pm 80^\circ/sec,\ \pm 120^\circ/sec]$
Integral p Error Gain $K_{I_{p-\delta_a}}$	-0.25
Integral β Error Gain $K_{I_{\beta-\delta_r}}$	0.25

5.2.2 Clean Data

The discrete model discovered from clean measurements, as in (5.4), is used in the MPC after the training and validation phases. A total of 70 seconds, showing all phases including the control phase, is shown in Figure 5.16. The reference input given in the control phase is shown with blue curves, and the actual response of the aircraft is shown with red curves.

In order to take a closer look at the responses in the control phase and to compare the responses of the classical controller and MPC, the results including only this 50-

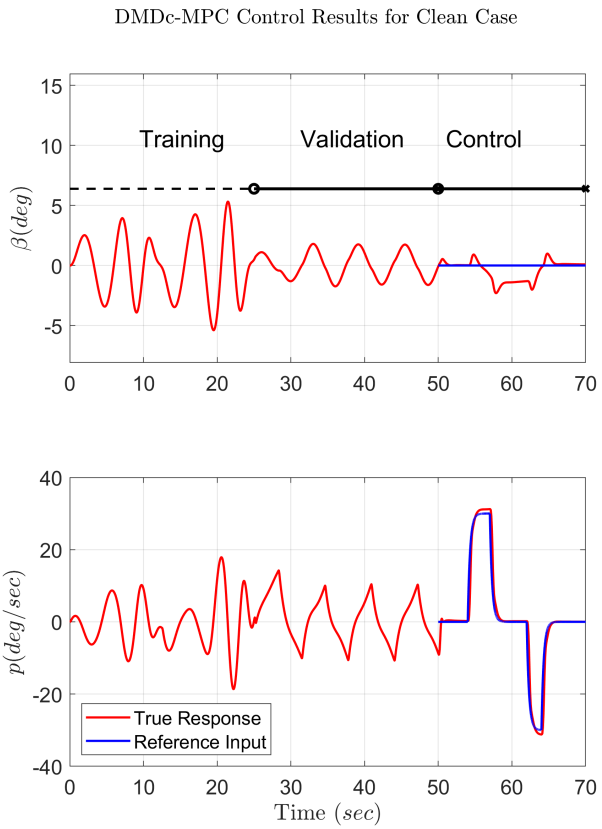


Figure 5.16: Output response for all phases for clean data

70 sec time interval are shown in Figure 5.17. In addition, in order to understand whether the sideslip due to roll and the roll mode time constant requirements given in Table 5.6 is met, the $\pm 2^{\circ}$ interval of β is shown with the green zone as the acceptable area and the time constant values of both controllers are written.

Control input position and rate graphs are shown in Figure 5.18 to understand the actuator activity. The reference control input values δ_{com} produced by MPC is shown with blue dashed curves. It can be observed from Figure 5.18 that the reference inputs δ_{com} produced by the MPC remain within the given constraints. In addition, the fact that the maximum position and rate values are not frequently reached can be interpreted as soft actuator usage for the MPC case.

The same graphs are drawn in Figure 5.19 for the Classic Controller case, and the blue dashed curves show the reference commands produced by the Classic Controller. It can be said that the results are very similar to the MPC case and the excessive actuator usage is not observed.

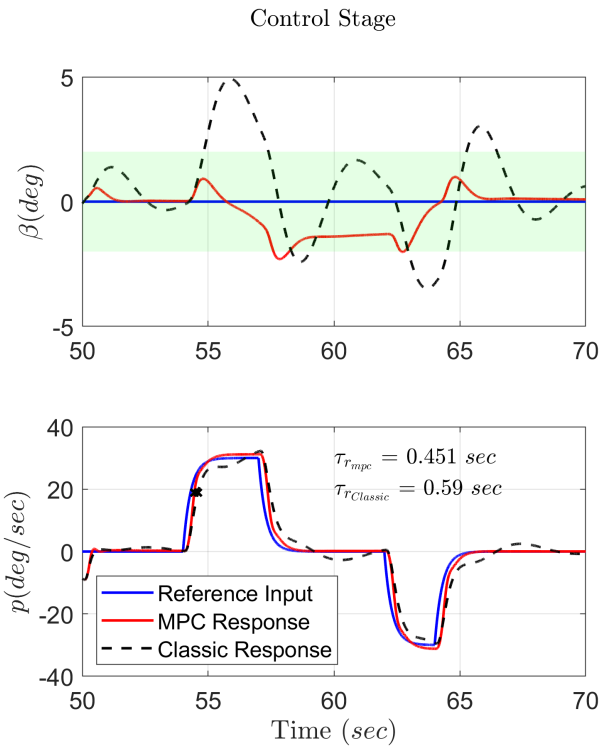


Figure 5.17: Output response for control phase for clean data

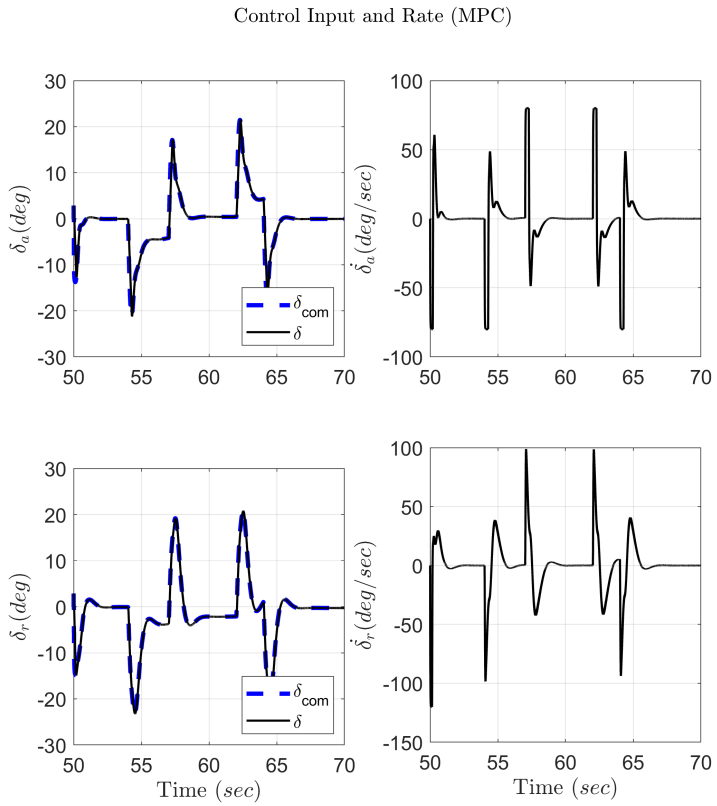


Figure 5.18: Actuator activity at control stage for MPC for clean data

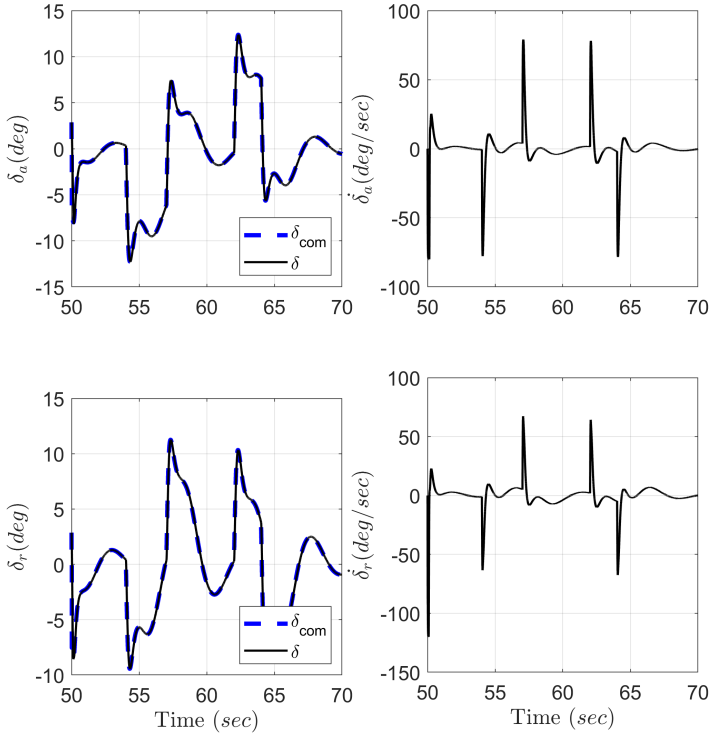


Figure 5.19: Actuator activity at control stage for Classic Controller for clean data

Robustness analyses are performed by adding additional gain and delays separately for each of the points specified in Figure 5.14 till the system becomes unstable. The GM and DM values obtained for each input channel are given in the Table 5.6. The results of the system becoming unstable with the added gain and delays are given in the Figure 5.20.

Actuator usage metrics are also calculated to compare the results numerically (see Table 5.8). The results show that the actuator usage of MPC and conventional controllers is quite different. In terms of rudder input δ_r CE and CRE values, the MPC is 10% and 95% higher than the Classic Controller, respectively. In terms of aileron input δ_a it is observed that MPC is 17% less for CE value and 68% higher for CRE value. It can be said that the MPC exhibited a more aggressive actuator usage for the clean measurement condition.

According to the results, the compatibility of the values obtained for the MPC and the classical controller cases with the given evaluation criteria are presented in Table 5.9 to understand whether they meet the requirements.

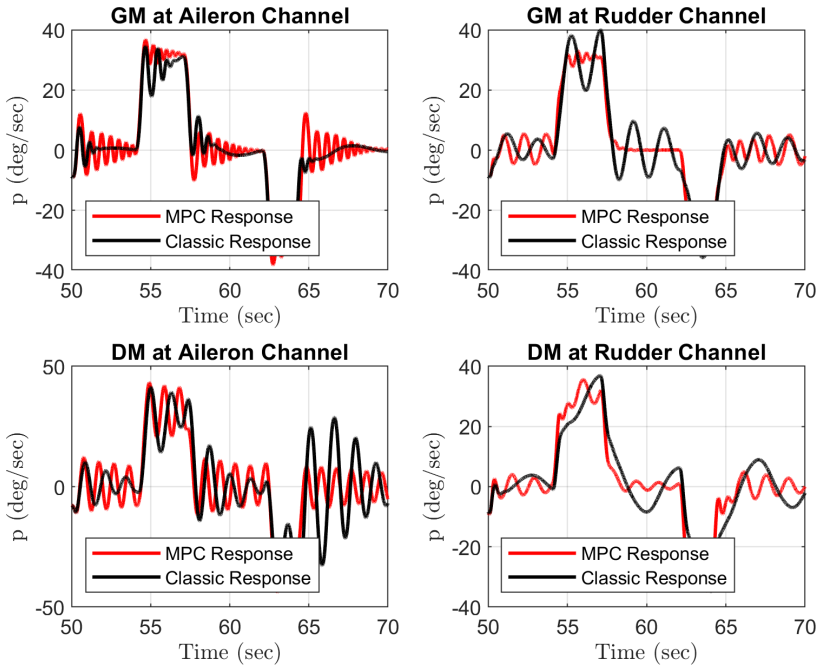


Figure 5.20: Responses for additional gain and delays

Table 5.8: Actuator usage metrics for the MPC and Classic Controller in the clean data case

Metric Name	MPC	Classic
CE for δ_r	82.9	74.39
CRE for δ_r	221.18	113.79
CE for δ_a	60.58	73.08
CRE for δ_a	183.50	109.67

Table 5.9: Design requirements compliance matrix for clean case

Requirement Name	Requirement	MPC	Classic
Single Loop GM at Aileron $[dB]$	GM ≥ 6	[-∞, 8.13]	[-∞, 8.13]
Single Loop GM at Rudder $[dB]$	GM ≥ 6	[-∞, 8.76]	[-∞, 9.54]
Single Loop DM at Aileron $[sec]$	DM ≥ 0.1	0.2	0.16
Single Loop DM at Rudder $[sec]$	DM ≥ 0.1	0.15	0.35
Effective Roll Mode Time Constant $[sec]$	$\tau_r \leq 1$	0.451	0.59
Maximum Roll Rate $[^\circ/sec]$	$\lvert p_{max}\rvert \geq 30$	≥ 30	≥ 30
Sideslip due to Roll $[^\circ]$	$\lvert\beta\rvert \leq 2$	≤ 2	≤ 5

It can be concluded that the MPC meets all the criteria, while the Classic Controller does not meet the sideslip due to roll requirement only.

5.2.3 Noisy Data

The control performance of MPC in noisy measurement situations is described in this section together with the classical control results.

5.2.3.1 Medium Noisy Data

The discrete model discovered from medium noisy measurements given in equation (5.8) is used in the MPC after the training and validation phases. A total of 70 sec showing all phases including the control phase is shown in Figure 5.21. The reference input given in the control phase is shown with blue curves, and the actual response of the aircraft is shown with red curves. The results in the 50-70 sec control phase window are shared in Figure 5.22. The obtained roll mode time constant values are indicated on the figure and the desired range for β is shown with the green area.

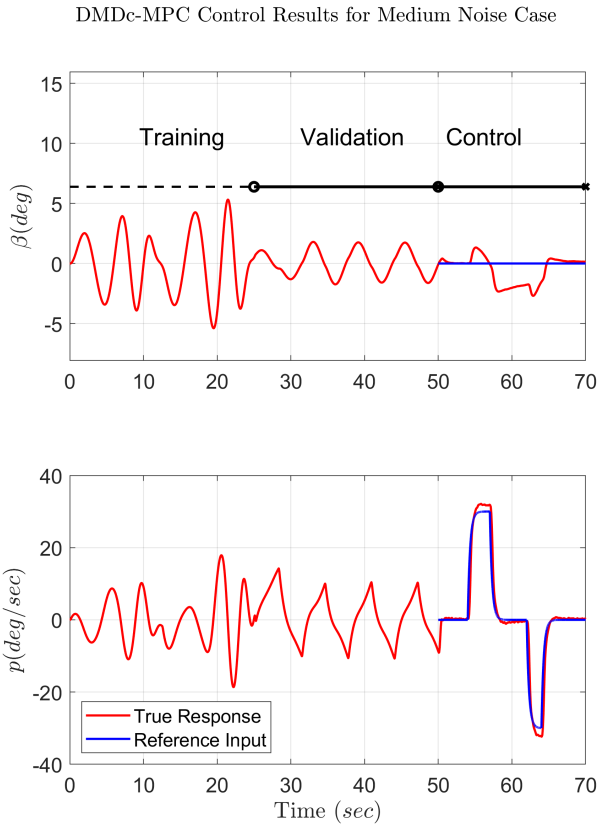


Figure 5.21: Output response for all Stages for medium noise

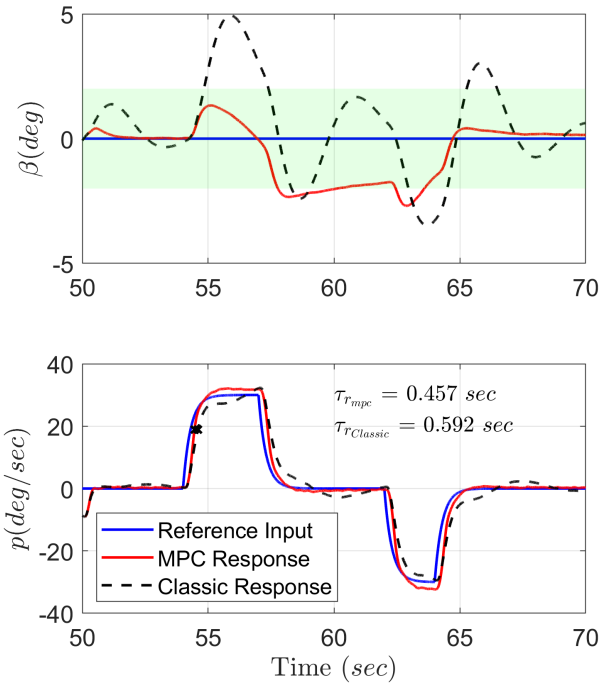


Figure 5.22: Output response for control stage for medium noise

Actuator activity during the control phase is shown in Figure 5.23. The reference control input values δ_{com} produced by MPC is shown with blue dashed curves. It can be observed from the Figure 5.23 that the reference inputs δ_{com} produced by the MPC remain within the given constraints. As can be seen from the figure, the control input is heavily affected by noise and the control input rates are very high.

In order to understand the actuator activity for the classical controller case, the actuator outputs are shown in Figure 5.24. The effect of medium noise on actuator activity also appears to be the same for the classical controller.

Actuator usage metrics are also calculated to compare the results numerically and given in Table 5.10. The results show that the CRE value is nearly doubled compared to the previous clean case for both MPC and classical controllers. It is observed that the MPC and classical controller results are close to each other in terms of CE values for both inputs. From the results, it can be said that medium noisy measurement greatly affects the actuator usage.

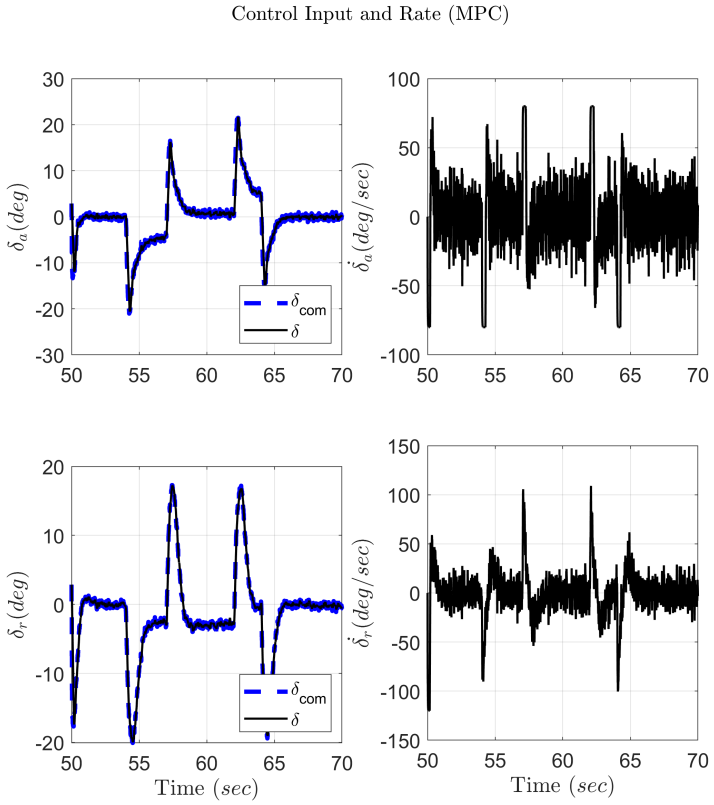


Figure 5.23: Actuator activity at control stage for MPC for medium noise

Table 5.10: Actuator usage metrics for the MPC and Classic Controller in the medium noise case

Metric Name	MPC	Classic
CE for δ_r	79.51	74.45
CRE for δ_r	298.48	217.21
CE for δ_a	65.40	73.15
CRE for δ_a	360.56	304.74

According to the results, the compatibility of the values obtained for the MPC and the classical controller with the given evaluation criteria and whether they meet the requirements are presented in Table 5.11.

Table 5.11: Design requirements compliance matrix for medium noise case

Requirement Name	Requirement	MPC	Classic
Effective Roll Mode Time Constant $[sec]$	$\tau_r \leq 1$	0.457	0.592
Maximum Roll Rate $[°/sec]$	$\lvert p_{max} \rvert \geq 30$	≥ 30	≥ 30
Sideslip due to Roll $[°]$	$\lvert \beta \rvert \leq 2$	≤ 2.5	≤ 5

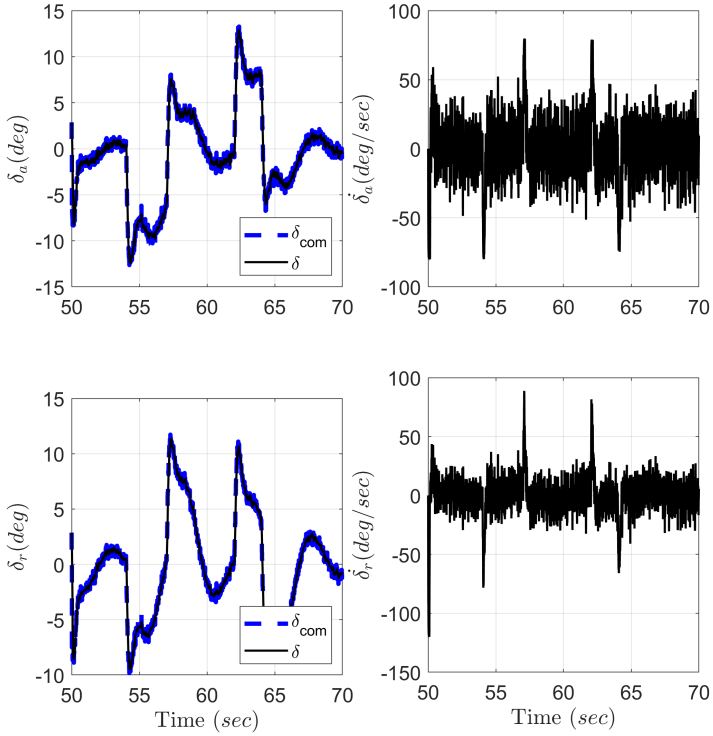


Figure 5.24: Actuator activity at control stage for classical controller for medium noise

Looking at the results, it can be concluded that the performance values of the Classic Controller are almost unchanged, but MPC cannot meet some requirements at this time.

5.2.3.2 High Noisy Data

The discrete model discovered from high noisy measurements given in equation (5.9) is used in the MPC after the training and validation phases. A total of 70 sec showing all phases including the control phase is shown in Figure 5.25. The reference input given in the control phase is shown with blue curves, and the actual response of the aircraft is shown with red curves.The results in the 50-70 sec control phase window are shared in Figure 5.26. The obtained roll mode time constant values are indicated on the figure and the desired range for β is shown with the green area.

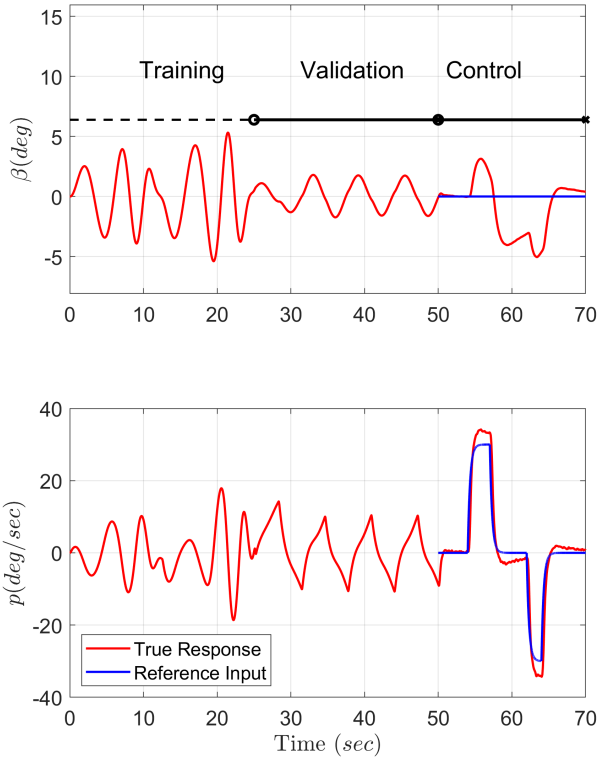


Figure 5.25: Output response for all stages for high noise

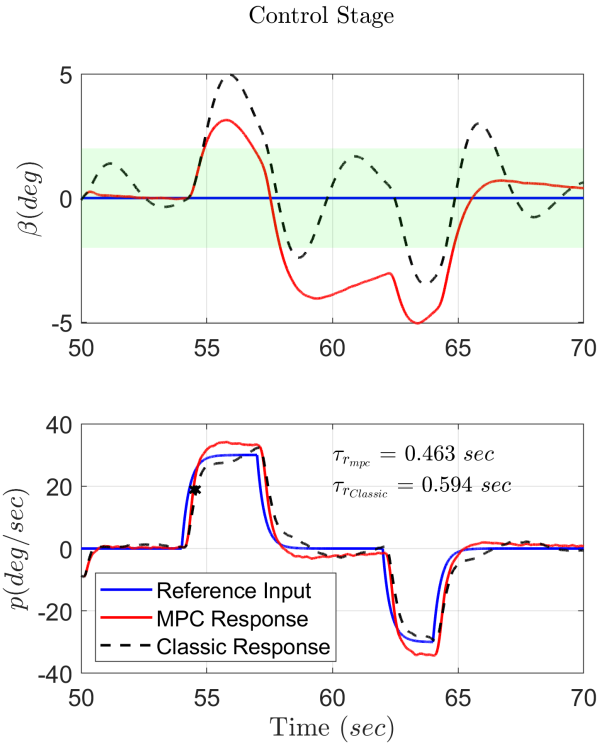


Figure 5.26: Output response for control stage high noise

Actuator activity during the control phase is shown in Figure 5.27. The reference control input values δ_{com} produced by MPC is shown with blue dashed curves. It can be observed from Figure 5.27 that the reference inputs δ_{com} produced by the MPC remain within the given constraints. As can be seen from the figure, the control input is heavily affected by high noise and the control input rates are higher than medium noise case.

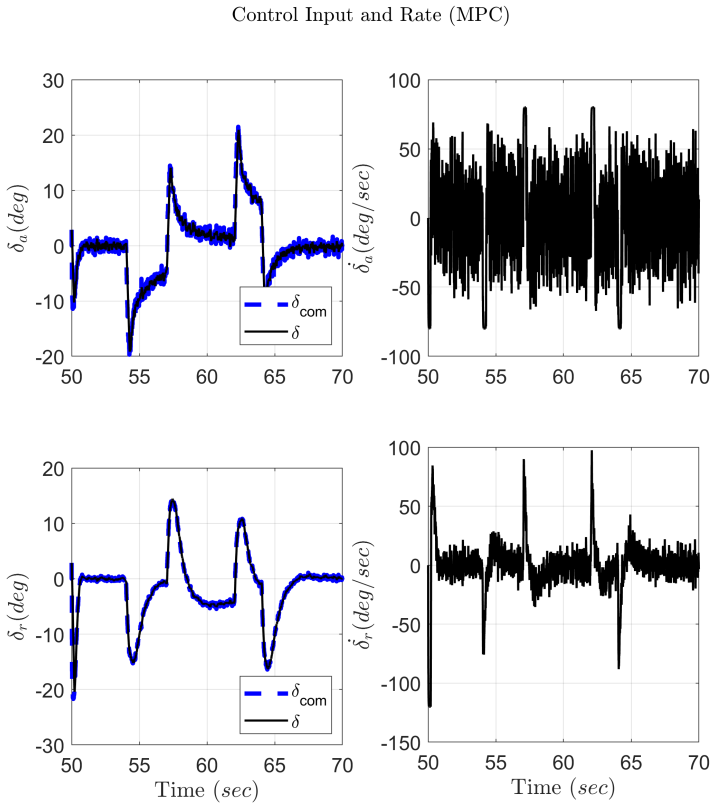


Figure 5.27: Actuator activity at control stage for MPC for high noise

In order to understand the actuator activity for the classical controller case, the actuator outputs are shown in the Figure 5.28. The effect of high noise on actuator activity also appears to be the same for the classical controller.

Actuator usage metrics are also calculated to compare the results numerically and given in Table 5.12. The results show that the CRE and CE values are increased compared to the medium noise situation for both MPC and Classical Controller. It is observed that the MPC and classical controller results are close to each other in terms of CE values for both inputs. From the results, it can be said that high noise affects the actuator usage more than the medium noise measurement situation.

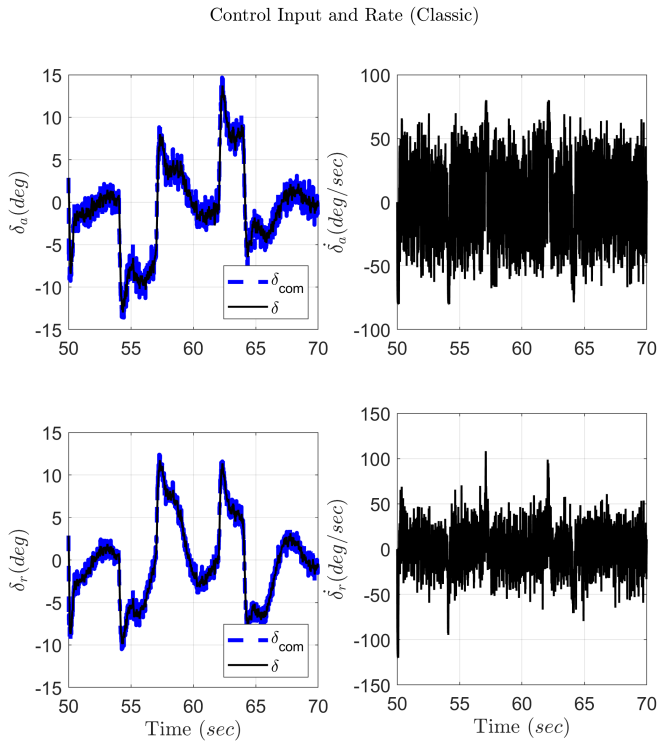


Figure 5.28: Actuator activity at control stage for classical controller for high noise

Table 5.12: Actuator usage metrics for the MPC and Classic Controller in the high noise case

Metric Name	MPC	Classic
CE for δ_r	78.89	74.72
CRE for δ_r	233.81	362.03
CE for δ_a	78.06	73.62
CRE for δ_a	471.2	485.65

According to the results, the compatibility of the values obtained for the MPC and the classical controller with the given evaluation criteria and whether they meet the requirements are presented in Table 5.13.

Table 5.13: Design requirements compliance matrix for high noise case

Requirement Name	Requirement	MPC	Classic
Effective Roll Mode Time Constant $[sec]$	$\tau_r \leq 1$	0.463	0.594
Maximum Roll Rate $[°/sec]$	$\lvert p_{max}\rvert \geq 30$	≥ 30	≥ 30
Sideslip due to Roll $[°]$	$\lvert\beta\rvert \leq 2$	≤ 5	≤ 5

Looking at the results, it can be concluded that the performance values of the Classic

Controller are almost unchanged, but MPC cannot meet some requirements at this time. In general, there is a decrease in MPC performance compared to the medium noise measurement case.

5.3 Real-time Computational Capability Validation Test

When the designed code will be used in a real application, it will run on a target hardware that does not contain Matlab/Simulink, unlike our desktop computer where the results of the previous sections are taken. The DMDc-MPC algorithm designed in Matlab/Simulink environment on the desktop computer is needed to be translated into C/C++ source code. Thanks to the Simulink Real-Time™ feature, the created Simulink model is converted to C/C++ code and automatically embedded in the target machine. The resulting C/C++ code is compiled with a specific target compiler for the processor and tested in the target hardware.

This test has two main purposes. The first main purpose is to check whether the compiled code also runs on the target processor. By comparing desktop and target processor simulation results, it can be tested that the numerical equivalence of Matlab/Simulink model and the generated code is achieved. The second main purpose is to analyze the computational cost of the DMDc-MPC algorithm. Since our daily used desktop computers do a lot of work in the background, this analysis is done in the most accurate way on a target hardware dedicated to this task.

Speedgoat unit real-time target machine is the ideal equipment for this test due to its fast testing and application capability. This equipment has a real time operating system from Mathworks which supports C/C++ source code compiler. The picture and specifications of the Speedgoat equipment used for this test are given in Figure 5.29 [1] and Table 5.14, respectively.

The real-time computational capability is validated in the following. Exploiting parallel computing using a multicore processor can increase the computational capabilities of our system [25]. For this reason, the option to execute tasks concurrently is selected and multicore architecture is used in the hardware configuration tab. In this experiment, the plant model is executed in core 1, while the MPC and DMDc algorithms

Figure 5.29: Speedgoat Unit Real-Time Target Machine [1]

Table 5.14: System Specifications

Manufacturer	Speedgoat®
CPU	Intel® Atom® 1.6 GHz quad-core
Main drive	120 GB SSD
Memory	4 GB DDR3
Software	Simulink Real-Time

are executed in cores 2 and 3. Both plant model, MPC and DMDc algorithms are synchronized by making the necessary rate transitions. Rate transition handling is done manually on the signals one by one. The sampling frequency of each core are given in Table 5.15. The multicore structure of the overall process is shown in Figure 5.30. The outputs of core 1 will be the command or reference for cores 2 and 3.

Table 5.15: Sampling Frequencies of Each Core

Cores	Sampling Frequency
Core-1 (Plant)	1000Hz(0.001s)
Core-2 (MPC)	100Hz (0.01s)
Core-3 (DMDc)	10Hz (0.1s)

5.3.1 Results

A real-time simulation is performed on Speedgoat to evaluate the real-time performance of DMDc-MPC algorithm. The signals to be logged are selected on the Simulink model and stored on the Speedgoat. These logs can then be accessed via Matlab with "*slrtexplorer*" and pulled into the workspace. The normal and target

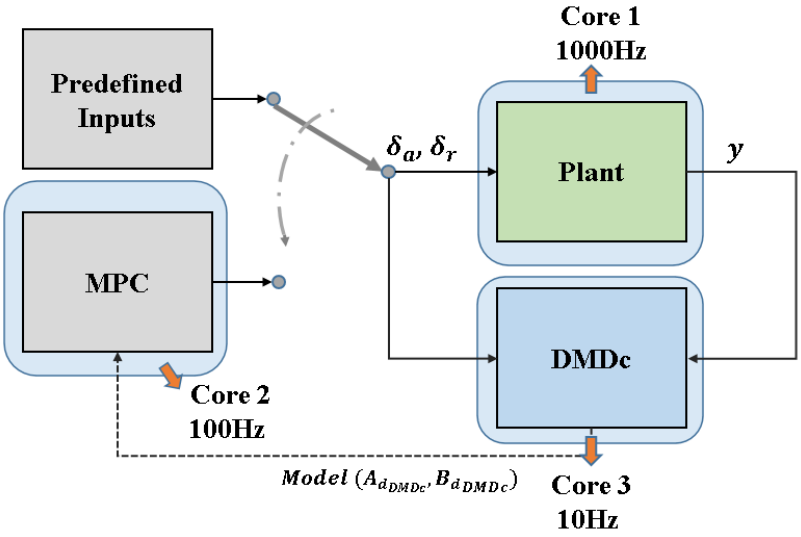


Figure 5.30: Multicore structure of Speedgoat.

hardware simulation results are compared to understand the consistency of the generated C/C++ code embedded in the target hardware with the desktop computer result. The comparison of the simulation results obtained on Speedgoat and the desktop computer is shown in Figure 5.31. In order to avoid a crowd of figures, only the comparative result of the roll rate response is shared. Results from Speedgoat are shown with blue dashed curves, and results from a desktop computer are shown with red curves.

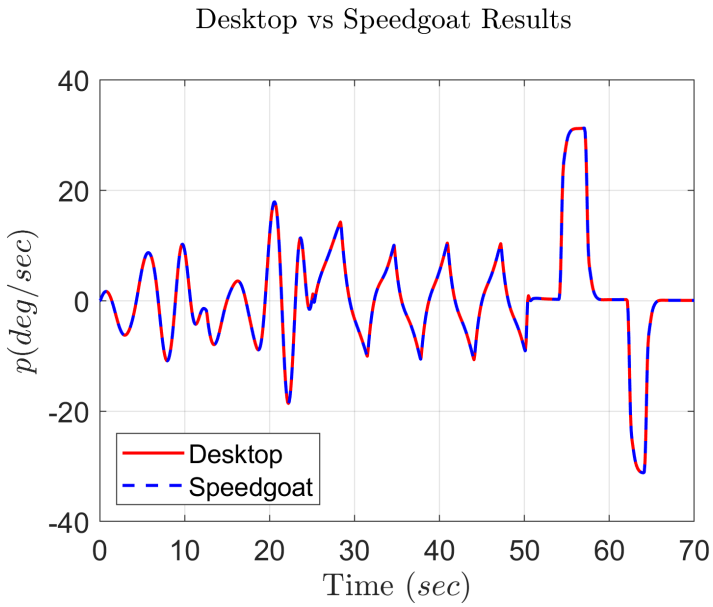


Figure 5.31: Normal vs Speedgoat Simulaton

The sampling frequency of the cores specified in Table 5.15 indicates how long it

will take to complete each iteration of the task to be performed. However, how long it takes for the task to actually be completed may differ from these values. For example, the task may have completed in a shorter time and the processor may have waited in sleep for this period without taking any action. The time it takes for the task to be completed is called the task execution time (TET). The variation of TET values for each core during the all simulation is given in Figure 5.32. Since DMDc creates the model at the end of the 25th second training period, we see a spike in the TET response given for core 3. On the other hand, we see a high increase in the TET response for core 2 since the control phase starts after the 50th second.

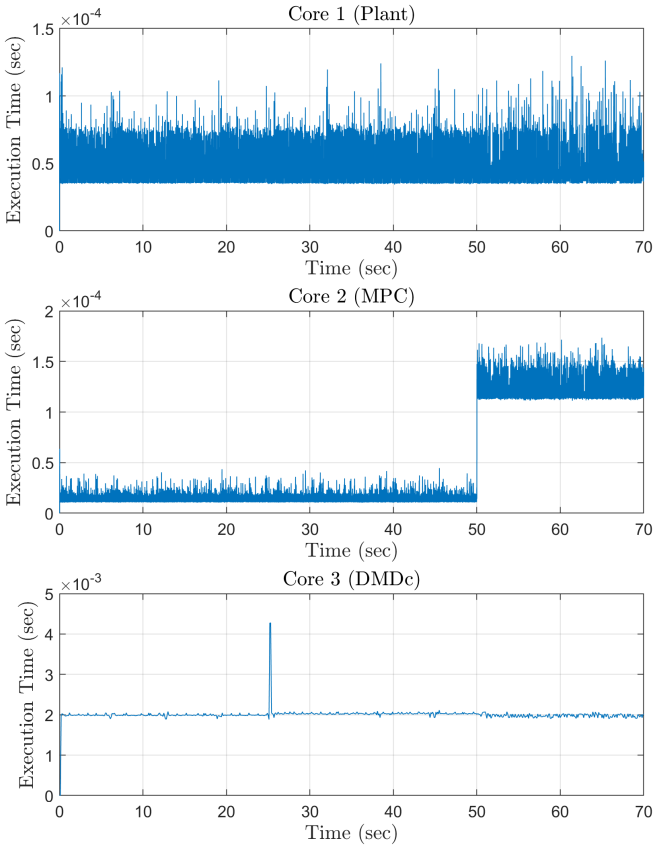


Figure 5.32: Target execution times for each cores during simulation

Defined sampling time for each core with the maximum and average values obtained from TET results are summarized in Figure 5.33. It can be understood that the task execution times are quite low compared to the sampling frequency defined for the cores. In summary, these results can be interpreted as the low computational cost of the DMDc-MPC algorithm.

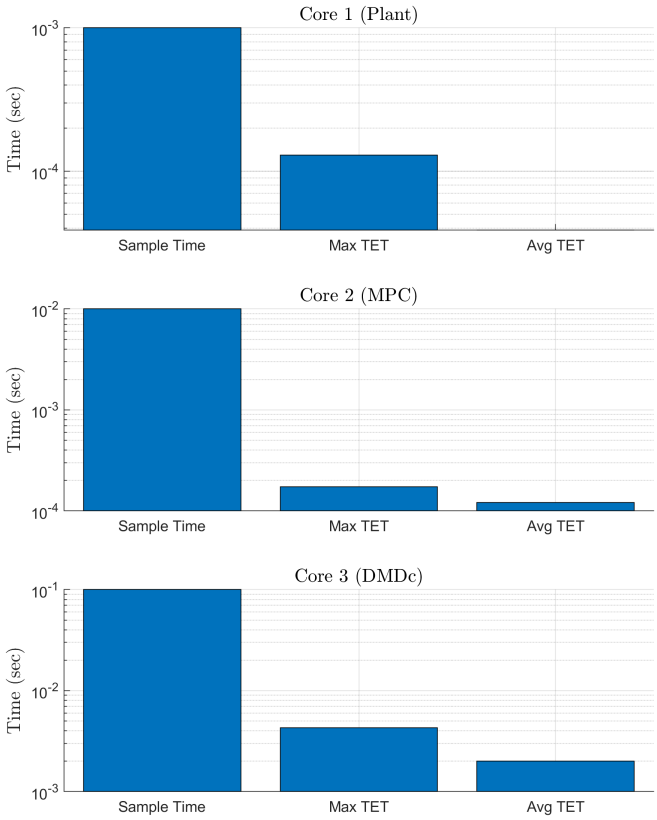


Figure 5.33: Summarized TET results

CHAPTER 6

CONCLUSION

It has been shown that the DMDc-MPC framework for control of lateral/directional fighter aircraft dynamics which is MIMO system, whose controller design is difficult using classical methods, is more successful than the Classical Controller in clean measurement situations. Although the better results can be obtained by optimizing the Classical Controller gains, the DMDc-MPC framework seems promising. It is observed that DMDc model discovery process is not robust against noisy situations but it has shown better results in control phase by meeting most of the evaluation criteria even in noisy measurement situations.

One of the main contributions of this study to the literature is to test the DMDc-MPC structure created on a MIMO system and obtain successful results. Another contribution is demonstrating the robustness of the DMDc-MPC structure against added delays and gains. The last and perhaps the most important contribution is to show the computational cost by testing the generated algorithm on the target hardware running in real time.

The efforts made in total can be summarized as follows: formulating the model discovery method and the QP problem for the control process properly, determining the QPKWIK algorithm to be used for solving the optimization problem efficiently, establishing a mathematical model of a fighter aircraft with its subsystems to test the resulting structure and lastly implementing the resulting algorithm in target hardware to show the real time computational capability.

In the future work, it can be studied that the generalization of the proposed algorithm

for other potential applications such as controlling robot motions can be studied.

This study also motivates several future investigations and extensions such as:

- If there are known terms in the dynamic, this information can be used to improve the DMDc model discovery accuracy.
- The precision of model discovery method DMDc can be tested with real flight test data.
- By testing the DMDc-MPC algorithm on an aircraft model with 6 degrees of freedom, it can be shown that successful results can also be obtained for high dimensional systems.
- Guaranteeing stability and robustness for MPC can be checked.
- In this study, it is assumed that the necessary measurements are available. But in reality, for example, the angle of sideslip signal may not be measured properly. Therefore, the DMDc-MPC structure can be matured by developing state estimation algorithms.
- It may be possible to develop an adaptive DMDc-MPC algorithm that will cover the entire flight regime by rapidly making new model discoveries according to flight conditions changes such as altitude and speed.
- Comparison can be made using different QP solver types to achieve a better computational cost.
- The computational cost can be shown by testing the algorithm on other units such as the GPUs.
- By changing the prediction horizon and control horizon values of the MPC, the increase in the computational cost and RAM requirement analysis can be made.
- Integral action was included in the MPC strategy to remove the steady state errors. The inclusion of integral gain in the optimization process can be considered instead of choosing fixed small values.
- The success of the DMDc-MPC algorithm can be proven by testing it on a real small scaled aircraft with a sufficient hardware.

REFERENCES

[1] Unit real-time target machine. https://www.speedgoat.com/products-services/real-time-target-machines/unit-real-time-target-machine, Accessed: 2022-06-07.

[2] H. Akaike. Fitting autoregressive models for prediction. *Annals of the Institute of Statistical Mathematics volume*, 21:243–247, 1969.

[3] S. Boyd, M. Hast, and K. J. Åström. MIMO PID tuning via iterated LMI restriction. *International Journal of Robust and Nonlinear Control*, 26(8):1718–1731, 2015.

[4] S. L. Brunton and J. N. Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019.

[5] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Compressive sampling and dynamic mode decomposition. 58:2426–2431, 2013.

[6] C. Buttrill and P. Arbuckle. *Simulation Model of a Twin-Tail, High Performance Airplane*. NASA, 1992.

[7] A. Chakraborty, P. Seiler, and G. J. Balas. Susceptibility of F/A-18 flight controllers to the falling-leaf mode: Linear analysis. *Journal of Guidance, Control, and Dynamics*, 34(1):57–72, 2011.

[8] M. Cook. *Flight Dynamics Principles*. Wiley, 1997.

[9] Y. Dong and J. Aii. Trial input method and ownaircraft state prediction in autonomous air combat. *Journal of Aircraft*, 49(3):947–954, 2015.

[10] Y. Dong, J. Tao, Y. Zhang, W. Lin, and J. Aii. Deep learning in aircraft design, dynamics, and control: review and prospects. *IEEE Transactions on Aerospace and Electronic Systems*, 57(4):2346–2368, 2021.

[11] A. Draeger, S. Engell, and H. Ranke. Model predictive control using neural networks. *IEEE Control Systems Magazine*, 15(5):61–66, 1995.

[12] C. Garcia, D. Prett, and M. Morari. Model predictive control: theory and practice — a survey. *Automatica*, 25:335–348, 1989.

[13] D. Goldfarb and A. Idnani. A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical Programming*, 27:1–33, 1983.

[14] M. Grilli, P. J. Schmid, S. Hickel, and N. A. Adams. Analysis of unsteady behaviour in shockwave turbulent boundary layer interaction. *J. Fluid Mech.*, 700:16–28, 2012.

[15] G. Guardabassi and S. Savaresi. Virtual reference direct design method: an off-line approach to data-based control system design. *IEEE Transactions on Automatic Control*, 45(5):954–959, 2000.

[16] H. Hjalmarsson, S. Gunnarsson, and M. Gevers. A convergent iterative restricted complexity control design scheme. *Proc. of the 33rd IEEE Conference on Decision and Control, Orlando, USA*, page 1735–1740, 1994.

[17] Z. Hou. The parameter identification, adaptive control and model free learning adaptive control for nonlinear systems. *PhD dissertation, Northeastern University, Shengyang, China*, 1994.

[18] Z. Hou and Z. Wang. From model-based control to data-driven control: Survey, classification and perspective. *Information Sciences*, 235:3–35, 2013.

[19] M. Hovd. A brief introduction to model predictive control. 2004.

[20] P. Huber and P. Seamount. X-31 high angle of attack control system performance. *Fourth High Alpha Conference*, 2(19), 1994.

[21] H. Jin, M. Stefanovic, and P. Tesi. Data-driven robust control: Special issue dedicated to the 70th birthday of Michael G. Safonov. *International Journal of Robust and Nonlinear Control*, 28(12):3665–3666, 2018.

[22] T. Johns. Should you be persuaded: Two samples of data-driven learning materials. *English Language Research, University of Birmingham*, pages 1–16, 1991.

[23] E. Kaiser, J. N. Kutz, and S. L. Brunton. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings of The Royal Society A Mathematical Physical and Engineering Sciences*, 474(14), 2018.

[24] R. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):3–35, 1960.

[25] J. Kepner. *Parallel MATLAB for multicore and multinode computers*. SIAM, 2009.

[26] B. O. Koopman. Hamiltonian systems and transformation in hilbert space. *Proc. Natl. Acad. Sci. USA*, 17:315–318, 1931.

[27] A. M. Kuethe and C.-Y. Chow. *Foundations of Aerodynamics: Bases of Aerodynamic Design*. Wiley, Michigan University, 1984.

[28] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor. *Dynamic mode decomposition: data-driven modeling of complex systems*. PA: SIAM, 2016.

[29] J. Lee. Model predictive control: review of the three decades of development. *Int. J. Control Autom. Syst*, 25:415–424, 2011.

[30] F. Lin, M. Fardad, and M. R. Jovanovic. Design of optimal sparse feedback gains via the alternating direction method of multipliers. *IEEE Trans. Automat. Control*, 58:2426–2431, 2013.

[31] J. Louw and H. Jordaan. Data-driven system identification and model predictive control of a multirotor with an unknown suspended payload. *Control Conference Africa : Magaliesburg, South Africa*, 54(21):210–215, 2021.

[32] J. L. M. Morari. Model predictive control: past, present and future. *Comput. Chem. Eng*, 23:667–682, 1999.

[33] P. Marion. Flipping the script with atlas. `http://bostondynamics.com/resources/blog/flipping-script-atlas`, Accessed: 2022-06-05, August 2021.

[34] H. Matpan. Data driven model discovery and control of longitudinal missile dynamics. *MSc thesis, Middle East Technical University, Ankara, Turkey*, 2021.

[35] D. Mayne. Model predictive control: recent developments and future promise. *Automatica*, 50:2967–2986, 2014.

[36] M. Millidere. Optimal input design and system identification for an agile aircraft. *PHd thesis, Middle East Technical University, Ankara, Turkey*, 2021.

[37] M. Mohsin. Model predictive control (MPC) with integral action; reducing the control horizon and model free MPC. *MSc thesis, Telemark University College, Norway*, 2013.

[38] NASA. *U.S. Standard Atmosphere*. National Oceanic and Atmospheric Administration, Washington, D.C., USA, 1976.

[39] L. T. Nguyen and M. E. Ogburn. *Simulator study of stall/post-stall characteristics of a fighter airplane with relaxed longitudinal static stability*. NASA, 1979.

[40] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, NY, USA, 2nd edition, 2006.

[41] R. Pratt. *Flight Control Systems: Practical Issues in Design and Implementation*. Institution of Electrical Engineers, 2000.

[42] J. Proctor, S. Brunton, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

[43] J. Proctor, S. Brunton, and J. N. Kutz. Dynamic mode decomposition with control. *SIAM*, 15:142–161, 2016.

[44] A. Punjani and P. Abbeel. Deep learning helicopter dynamics models. *IEEE International Conference on Robotics and Automation (ICRA)*, 20:3223–3230, 2015.

[45] M. S. Roeser and N. Fezans. Method for designing multi-input system identification signals using a compact time-frequency representation. *CEAS Aeronautical Journal*, 12:291–306, 2021.

[46] C. W. Rowley, I. Mezic, S. Bagheri, P. Schlatter, and D. S. Henningson. Spectral analysis of nonlinear flows. *J. Fluid Mech.*, 641:115–127, 2009.

[47] C. Schmid and L. T. Biegler. Quadratic programming methods for reduced hessian sqp. *Computers chemical engineering*, 18(9):817–832, 1994.

[48] P. J. Schmid, K. E. Meyer, and O. Pust. Dynamic mode decomposition and proper orthogonal decomposition of flow in a lid-driven cylindrical cavity. *the 8th International Symposium on Particle Image Velocimetry*, 2008.

[49] P. J. Schmid and J. L. Sesterhenn. Dynamic mode decomposition of numerical and experimental data. *Bull. Amer. Phys. Soc. 61st Annual Meeting of the APS Division of Fluid Dynamics (San Antonio, TX)*, 53:208, 2008.

[50] R. Skelton. Model error concepts in control design. *International Journal of Control*, 49(5):1725–1753, 1989.

[51] R. Stengel. *The small angle approximation. In The Ray and Wave Theory of Lenses (Cambridge Studies in Modern Optics, pp. 81-86)*. Princeton University Press, 2004.

[52] B. L. Stevens, F. L. Lewis, and E. N. Johnson. *Aircraft Control and Simulation*. John Wiley Sons, 2015.

[53] J. Terlouw. *Robust Flight Control Design Challenge Problem Formulation and Manual: The High Incidence Research Model*. GARTEUR, 1996.

[54] E. U, A. Prach, B. Koçer, S. Rakovic, E. Kayacan, and B. Açıkmeşe. Model predictive control in aerospace systems: current state and opportunities. *J. Guid. Control Dyn.*, 40:1541–156, 2017.

[55] M. Uchiyama. Formulation of high-speed motion pattern of a mechanical arm by trial. *Transactions of the Society of Instrument and Control Engineers*, 14(6):706–712, 1978.

[56] USAF. *MIL-1797-A: Military Standard, Flying Qualities of Piloted Vehicles*. Washington, D.C., USA, 1990.

[57] A. Walther. *The small angle approximation. In The Ray and Wave Theory of Lenses (Cambridge Studies in Modern Optics, pp. 81-86)*. Cambridge University Press, Cambridge, 1995.

[58] T. Zhang, G. Kahn, S. Levine, and P. Abbeel. Learning deep control policies for autonomous aerial vehicles with MPC-guided policy search. *IEEE International Conference on Robotics and Automation, Stockholm, Sweden*, 20:528–535, 2016.

APPENDIX A

In order to understand the effect of different data sampling times on model discovery, the mean errors between the eigenvalues of the discovered and real model are plotted in A.1 by performing offline analyzes. Algorithm that is used to calculate Hessian matrix $\mathbf{H}$ and related matrices, $\mathbf{K_r}$, $\mathbf{K_u}$ and $\mathbf{K_x}$ for cost function formulation is given in Listing A.1.

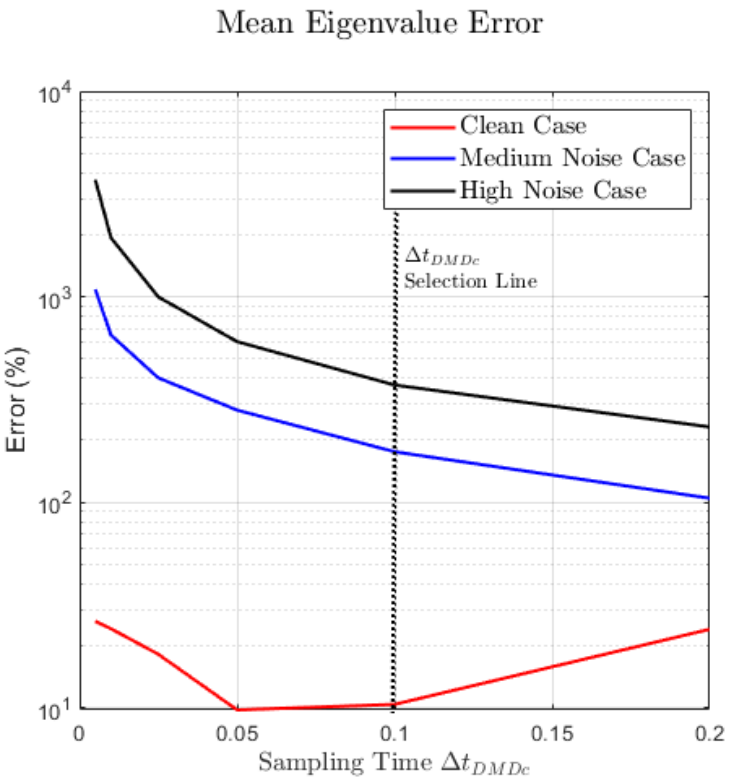


Figure A.1: MEE vs Sample rate $\boldsymbol{\Delta_{DMDc}}$

```
function [H,Ku1,Kx,Kr] = Hessian(A,B,C,Wy,Wdu,p,m)
% Calculate Hessian and related terms for controller initialization
% Author:  Can Öznurlu
% Inputs:  A, B, C: prediction model
%          Wy/Wdu are weight vectors
%          p/m are prediction and control horizon
ny = size(C,1);      % number of OV
nu = size(B,2);     % number of MV
pny1 = (p-1)*ny;
mnu = (m)*nu;
mnu1 = (m-1)*nu;

% Initialization and pre-allocation
CA  = C*A;
Sum = C*B;
Sx  = [CA; zeros(pny1,ny)];
Su1 = [Sum; zeros(pny1,nu)];
Su  = [Sum, zeros(ny,mnu1); zeros(pny1,mnu)];
Q   = [Wy';zeros(pny1,1)];
R   = [Wdu';zeros(mnu1,1)];

% Loop through horizon
for i=2:p
    rows = (i-1)*4+(1:4);
    rows_u = (i-1)*2+(1:2);
    Sum = Sum + CA*Br;
    Su1(rows,:) = Sum;
    CA = CA*Ar;
    Sx(rows,:) = CA;
    Q(rows,1) = Wy';
    % Loop through control horizon
    if i <=m
    R(rows_u,1) = Wdu';
    Su(rows,:) = [Sum, Su(rows-ny,1:(m-1)*nu)];
    else
    end
end

% Compute cost function Hessian and linear terms
H   =  Su'*diag(Q)*Su + diag(R) ;
Kr  = -diag(Q)*Su;
Ku1 =  Su1'*diag(Q)*Su ;
Kx  =  Sx'*diag(Q)*Su;
```

Listing A.1: Calculation of Hessian and related terms