AN EFFICIENT IMPLEMENTATION OF ONLINE MODEL PREDICTIVE CONTROL WITH PRACTICAL INDUSTRIAL APPLICATIONS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

OKAN ARPACIK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

AUGUST 2021

Approval of the thesis:

AN EFFICIENT IMPLEMENTATION OF ONLINE MODEL PREDICTIVE CONTROL WITH PRACTICAL INDUSTRIAL APPLICATIONS

submitted by **OKAN ARPACIK** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. İlkay Ulusoy
Head of Department, **Electrical and Electronics Engineering** _____

Assist. Prof. Dr. Mustafa Mert Ankaralı
Supervisor, **Electrical and Electronics Engineering, METU** _____

Examining Committee Members:

Prof. Dr. Kemal Leblebicioğlu
Electrical and Electronics Engineering, METU _____

Assist. Prof. Dr. Mustafa Mert Ankaralı
Electrical and Electronics Engineering, METU _____

Prof. Dr. Umut Orguner
Electrical and Electronics Engineering, METU _____

Assoc. Prof. Dr. Ozan Keysan
Electrical and Electronics Engineering, METU _____

Assist. Prof. Dr. İsmail Uyanık
Electrical and Electronics Engineering, Hacettepe University _____

Date:04.08.2021

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, Surname: Okan Arpacık

Signature :

ABSTRACT

AN EFFICIENT IMPLEMENTATION OF ONLINE MODEL PREDICTIVE CONTROL WITH PRACTICAL INDUSTRIAL APPLICATIONS

Arpacık, Okan
M.S., Department of Electrical and Electronics Engineering
Supervisor: Assist. Prof. Dr. Mustafa Mert Ankaralı

August 2021, 81 pages

The demand to utilize modern control algorithms for industrial applications is much more intensive. Model-predictive-controller (MPC), which is one of the modern optimal control policies, has gained more attention in servo drive and other industrial applications in recent years due to increased computational capabilities of embedded platforms and evident control performance benefits compared to more classical control methods. A digital MPC algorithm at each sampling instant produces the optimal control input sequence for a given prediction horizon while also guaranteeing that input and state-trajectories do not violate some set of constraints. Its optimization based capability brings more flexibility to include the additional requirements such as energy efficiency, quality of the systems' control input. Solving constraint optimization problems in each step requires excessive computational complexity and burden, which is the main drawback of online MPC over classical methods. In this thesis, we demonstrate the feasibility of online MPC in high sample frequency applications and provide some suggestions for practical implementation. We implemented the existing dual active set solver by replacing two common methods in the matrix update step to increase the performance in terms of execution speed. We also provide the linear

approximation for the nonlinear constraints by taking the tradeoff between accuracy and speed into account. The proposed structure is successfully verified via both PIL simulation and experimental testing. In addition, two different processors, which are commonly used in motion control applications, perform the PIL simulation and experimental testing separately to certify the feasibility of our implementation in terms of execution speed and using minimal memory space.

Keywords: Active-Set Method, Constraints, Field weakening, Gimbal Platform, Model Predictive Control(MPC), Quadratic Programming (QP), Real-time Optimization, Synchronous Machine

ÖZ

ÇEVRİMİÇİ MODEL ÖNGÖRÜLÜ KONTROLÜN PRATİK ENDÜSTRİYEL UYGULAMALARIYLA ETKİLİ BİR UYGULAMASI

Arpacık, Okan
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Dr. Öğr. Üyesi. Mustafa Mert Ankaralı

Ağustos 2021 , 81 sayfa

Endüstriyel uygulamalarda modern kontrol algoritmalarının uygulanma talebi hiç olmadığı kadar fazladır. Modern kontrol metotlarından biri olan Model Öngörülü Kontrole(MPC), gömülü platformların hesaplama kabiliyetlerindeki artış ve klasik kontrol metotları üzerinde içermiş olduğu yeteneklerden ötürü son yıllarda servo ve endüstriyel uygulamalarında ilgi artmıştır. Sayısal MPC algoritması, her bir hesap adımında belirlenen öngörü penceresinde sistemdeki kısıtları da sağlayacak şekilde optimal kontrol girdisini üretmektedir. Optimizasyon tabanlı yapısı, sistemler için ek isteklerin örneğin; enerji verimliliği veya kontrol işaretinin kalitesi, eklenmesinde daha fazla esneklik sağlamaktadır. Çevrimiçi MPC'nin en büyük dezavantajı olan her bir hesap adımında kısıtlı optimizasyon problemini çözmek çok fazla işlem gereksinimine ihtiyaç duymaktadır. Bu tezde, çevrimiçi model öngörülü kontrolün yüksek örnekleme frekanslarında uygulanabilirliğini gösterdik ve pratik uygulamalar için öneriler sunduk. Mevcut ikili aktif küme metodunu matris güncelleme adımında bilinen iki matris metotlarıyla değiştirerek performansı çalışma hızı bakımından artırdığımızı gösterdik. Ayrıca, doğrusal olmayan kısıtlamalar için hız ve doğruluğu da dikkate alarak

doğrusal yaklaşımlar gerçekleştirdik. Önerilen yaklaşım hem PIL simülasyonu hem de deneysel testlerle başarılı bir şekilde doğrulandı. Buna ek olarak, uyarlamamızın hem çalışma hızı hem de az hafıza kullanımı bakımından uygulanabilirliğini onaylamak için PIL simülasyonu ve deneysel testler hareket kontrol uygulamalarında çokça kullanılan iki farklı işlemciler ile gerçekleştirildi.

Anahtar Kelimeler: Aktif Küme Methodu, Kısıtlar, Alan Zayıflatma, Gimbal Platform, Model Öngüörülü Kontrol, Karesel Programlama , Gerçek-zamanlı Optimizasyon, Senkron Sakine

To my family and my beloved ones,

ACKNOWLEDGMENTS

First of all, I would like to express my sincere feelings about my supervisor M. Mert Ankaralı for his support and guidance on taking the right actions during my thesis period. I am thankful for having the opportunity under his supervision that lead me to complete my thesis successfully.

I feel fortunate for having innovative, talented colleagues at Aselsan Inc. that help to increase my engineering strength and gain perspectives on how to look at a problem to resolve. I would like to thank those people, starting from M. Burak Gürcan. Other colleagues, Ercan Çandır and Umut Gökkaya always support me to never lose my motivation and belief. I would like to open an extra bracket to Aykut Demirel for his modest personality and assistance to my works.

I would like to acknowledge Aselsan Inc., which provides hardware setup and physical platforms that make the implementation phase of this thesis easier.

I owe the special thanks to my friends, especially Onur Karahan, Ömer Herekoğlu, Fatih Karaca, and Çevikalp Sütunç, who have always walked with me no matter what happens and never spare their support on me.

I reserved to final statements for my great family. I gain my self-reliance thanks to their endless and limitless supports. They always gave me the power to overcome the problems that I encountered in tough times.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

ADC	Analog to Digital Converter
AS	Active Set
CCS	Continuous Control Set
DAC	Digital to Analog Converter
EMF	Electro-motor Force
FCS	Finite Control Set
FOC	Field Oriented Control
FW	Field Weakening
ISE	Integral Square Error
KKT	Karush Kuhn Tucker
LTI	Linear Time Invariant
MPC	Model Predictive Control
MP-TC	Model Predictive Torque Control
MTPA	Maximum Torque Per Ampere
PID	Proportional Derivative Integral
PIL	Processor In the Loop
PMSM	Permanent Magnet Synchronous Motor
PWA	Piece-Wise Affine
PWM	Pulse Width Modulation
RHC	Receding Horizon Control
QP	Quadratic Programming
VSI	Voltage Source Inverter

CHAPTER 1

INTRODUCTION

1.1 The Motivation and Scope of the Thesis

The demand from control algorithms has reached beyond the capability of conventional or classical techniques. Although the classical approaches such as PID proved their reliability in many applications, the control algorithms evolved in such a way to meet the requirements of modern control applications. Modern control methods such as predictive control for industrial applications have been getting more attention to control such systems thanks to their features over classical approaches. Like other optimal control techniques, MPC derives optimal control signals to the system according to the pre-defined cost function. What makes the MPC different from the other optimal control algorithms is to handle the system's constraints explicitly. The clarity of MPC is to use the model of the process to make predictions on the future evolution of the plant. It basically produces the optimal control input by taking the constraints and the future system behaviors into consideration. This technique is not only applicable to the special classes of the system but many from aerospace to automotive and still growing to extend to many other areas.

In addition to tracking phenomena, which is the minimum requirement from the control algorithm, modern industrial applications need to utilize energy efficiency, exploit the driver unit to its maximum capacity and produce the optimal control signals by taking all these requirements into account. The idle speed control or the temperature control in the vehicle are only two examples for such systems where the fuel efficiency is included in the control algorithms as an additional requirement. On the other hand, such algorithms usually bring more computational complexity and bur-

den since it solves an optimization problem to produce the optimum control inputs to the system. However, as the computational power of embedded platforms evolved, the adoption of MPC in a wide range of control applications, including but not limited to high-bandwidth plants and challenging performance specifications, have also increased in the past two decades [3].

In the literature, studies have integrated MPC-type algorithms for the control of electrical drive systems where the computational demand is higher. Several studies adopted online optimization-based MPC algorithms for regulating torque output of PMSMs (MP-TC). Like our motivation in this thesis, most of the studies implemented MPC in a "high-performance" processor and consuming a large amount of memory space. The authors also intended to improve their methodology to make it implementable with a low-cost embedded platform with far less CPU power and memory space in the future work section of their works.

Implementing MPC in low-cost processors, which are widely used in industrial applications, boosts us to investigate this application. The challenge of reducing the computational complexity in the online optimization problem with sample time in order of μs is another motivation source. We chose to control the PMSM as our first example that dominates most of the industrial servo applications due to its long-term usage, power density, and torque-speed characteristic. We replace the MPC in place of two different PI controllers for currents loops in the FOC algorithm. Thus, we demonstrate the practical feasibility of our implementation in a high bandwidth system. Model-based approaches increase the quality of the controller for motion control applications by taking the model-based nonlinear effects in the systems into account. MPC is a suitable alternative for other model-based approaches thanks to its natural structure, allows to include the effects in the system, and has the ability to produce optimal control input while respecting the dominant undesirable effects in the system. MPC might overcome the disturbances that affect the system performance by including them in the model if they are measurable or adding integral action in the MPC if they are unmeasurable. As our second example, we implement MPC to control the one axis of the gimbal platform. We provide some additional information about dealing with the delay term in the system that most practical industrial applications suffer from.

In this thesis, we implement the MPC with practical industrial applications, i.e., field weakening operation in PMSM, which requires fast sample frequency and controlling the gimbal platform to deal with friction and take the delay in the system into account.

1.2 The Outline of the Thesis

This thesis is organized to give the mathematical background for MPC and Active Set Solver and their implementation on industrial applications. Throughout the thesis, we provide some practical implementation suggestions for the readers.

- In the first chapter of the thesis, we provide some mathematical preliminaries for further usage and describe our motivation and scope of this work.
- In the second chapter, we describe the mathematical background for model predictive control. The problem formulations to meet different requirements and their condensed form(after casting QP) are provided.
- In the third chapter, we present the online optimization in MPC, first introducing the unconstrained case. Then, the quadratic programming that depends on the dual active set solver is expressed for its two methods: primal and dual. We provide implementation details of efficient matrix updating strategy in dual active set solver as the proposed algorithm. Finally, we evaluate our approach in the two common literature examples by comparing them in terms of executing speed.
- In the fourth chapter, we implement the MPC in a widely used industrial motor: PMSM. We provide the mathematical model and motor model verification procedure for PMSM. The controller design procedure for multi-variable systems and linear approximation for nonlinear constraints are also provided. We firstly evaluated the field weakening operation under PIL simulations to ensure that our implementation is feasible. Then, we tested our algorithm as a final step in custom-made PMSM under a laboratory environment.
- In the fifth chapter, we implement the MPC in the gimbal platform to control the traverse axis of the platform, for which we eliminated the effect of the fric-

tion by including a linear approximated friction model in the MPC. The performance of MPC for the gimbal platform is evaluated via physical experiments.

- In the last chapter of this thesis, we conclude by discussing the results and providing our plans for future works.

1.3 Preliminaries

Before introducing the basics of the MPC, we will provide some basic mathematical definitions in this section of the thesis for the ease of readers. The definitions are the basis for the mathematical background of quadratic optimization and the active set method. Further information is available in the textbooks [4, 5].

Definition 1.3.1 *(Convex Set) A set $S \subset \mathbb{R}^n$ is said to be convex if line segment any two points in the set S lies completely in the set. S is said to be convex set if $\forall \bar{x_1}, \bar{x_2} \in S$, we have*

$$\lambda \bar{x_1} + (1-\lambda)\bar{x_2} \in S, 0 \leq \lambda \leq 1 \tag{1.1}$$

Definition 1.3.2 *(Convex Function) Let $f : S \to \mathbb{R}$ is non-empty convex set in $\mathbb{R}^n$, f is said to be convex function if*

$$f(\lambda \bar{x_1} + (1-\lambda)\bar{x_2}) \leq \lambda f(\bar{x_1}) + (1-\lambda) f(\bar{x_2}) \tag{1.2}$$
$$\forall \bar{x_1}, \bar{x_2} \in S$$
$$0 \leq \lambda \leq 1$$

Definition 1.3.3 *(Quadratic Programming) An optimization problem is said to be quadratic programming if the convex objective function is in the quadratic form and it is subject to linear constraints,*

$$\min_z \ \frac{1}{2} z^T H z + z^T g \tag{1.3a}$$

subject to

$$Wz \leq b$$
$$W_e z = b_e \tag{1.3b}$$

where $H \in \mathbb{R}^{nxn}_{\geq 0}$, $g \in \mathbb{R}^{nx1}$,$W \in \mathbb{R}^{mxn}$, $b \in \mathbb{R}^{m}$, $W_e \in \mathbb{R}^{m_e xn}$, $b_e \in \mathbb{R}^{m_e}$.

Definition 1.3.4 *(Active Set) For the problem in(1.3), Active Set is defined as to include equality constraints,i.e.,*

$$W_{\mathbb{A}}(\tilde{z}) = \{i \in k | W_i \tilde{z} = b_i\} \tag{1.4}$$

where $k = \{1, 2, \ldots, n\}$ *is set of the constraint indexes and* $W_{\mathbb{A}}(\tilde{z})$ *is active set.*

Definition 1.3.5 *(KKT Conditions) For a solution of optimization problem to be optimal, first-order necessary condition, also known as Karush-Kahn-Tucker(KKT) conditions, must be satisfied. Given a problem,*

$$\begin{aligned} &\textit{minimize } f(z) \\ &\textit{subject to: } g(z) \leq 0 \\ &h(z) = 0 \end{aligned} \tag{1.5}$$

the conditions for this problem expressed as

$$\begin{aligned} \nabla f(z) + \sum_{i=1}^{m} \lambda_i \nabla g_i(z) + \sum_{j=1}^{r} \mu_j \nabla h_j(z) = 0 \quad &, \textit{stationarity} \\ g(z) \leq 0 \quad &, \textit{primal feasibility} \\ h(z) = 0 \quad & \\ \lambda \geq 0 \quad &, \textit{dual feasibility} \\ \forall i \in 1, \ldots, m \lambda_i g_i(z) = 0 \quad &, \textit{complementary slackness} \end{aligned} \tag{1.6}$$

CHAPTER 2

MODEL BASED PREDICTIVE CONTROL

Model predictive control (MPC) is a modern control policy family widely used for controlling systems while satisfying a set of constraints. Its relatively simple structure, ability to satisfy the constraints on both inputs and states, explicitly handling the multivariable system, and taking the model itself into consideration to generate optimal control sequence to the system are the significant features of the model predictive control[6, 7]. MPC solutions are also easier to adapt to different applications and provide great scalability in large-scale applications where the same control strategy needs to be applied to different versions of the same application. One of the most (if not the most) popular feedback control strategies that can enforce such constraints while also ensuring other critical system properties such as stability is the constrained MPC. It is a powerful tool in the sense that it estimates the future behavior of the system and generates the optimal input at each time step.

Several designations such as Generalized Predictive Control(GPC), Receding Horizon Control(RHC)[8] have been made to replace the name of MPC since the first use of the MPC idea in the late 70s. The early implementation of MPC was utilized to fulfill the requirements in the petrochemical industry, where the required sample time is slow. Shell Oil Engineers developed the first reported algorithm as the name of Dynamic Matrix Control with the model derived from input-output experiments[9]. Since then, technological progress in the chip industry enables the utilization of such algorithms in systems with faster dynamics, e.g., from automotive to aerospace. The usage of MPC in systems with much faster dynamics also has gained more attraction over the past two decades[3]. Nowadays, it is even possible to see MPC implementations in the control of power electronics and electrical drives where sampling

frequency requirements are several orders of magnitude higher than ones used in the initial deployments of MPC [10, 11].

Even though MPC provides a solid framework for advanced and high-performance control system design, it brings some undeniable disadvantages, precisely extra computational burden and increased implementation complexity, over dominantly adopted classical approaches. In that respect, researchers in the MPC domain mainly focus on solving computational and implementation complexity problems. Bemporad et al. [12, 13] proposed the explicit quadratic regulator for constrained systems which technically pushes the majority of the computations of the MPC algorithm to an offline pre-processing phase that increased the applicability and popularity of the MPC in experimental and industrial platforms. Specifically, this method generates a lookup table of Piece-wise Affine (PWA) functions that can radically reduce the execution time compared to online MPCs. On the other hand, memory requirements can grow dramatically with the dimension of the system and constraints. In addition to being a memory-dependent type of controller, the core advantage of the algorithm tends to break down with changes in the system model, such as physical parameters, which could require the repetition of the whole offline pre-processing phase.

2.1 Receding Horizon Control

MPC is nearly universally implemented as a digital/discrete control. It generates an input sequence at each sampling instant for the indicated horizon to minimize the related cost function while respecting the specified constraints. The idea of calculating the manipulated signals online is commonly referred to as Receding Horizon Control. In order to generate the optimal manipulated signal, a finite horizon optimal control problem shall be solved in each sample according to the desired performance criteria. MPC aims to find an optimal input signal that will drive the system to the desired configuration by minimizing a quadratic cost function. The tracking of the desired set-point is the minimum criteria to be fulfilled. Fig.2.1 illustrates the idea of the receding(shifting) horizon control over the predictions on state and manipulated variables. H_p determines the maximum number of future responses from time k., the parameter H_u is referred to as the parameter for the prediction of the number of free

moves. There is a degree of freedom to select the H_u parameter independently of the H_p, and it is commonly chosen as $H_u < H_p$. The values of manipulated signals from H_u to H_p are holding as constant. Taking the H_u parameter less than H_p helps to reduce the computational burden since the optimization dimension is determined by H_u. Once MPC calculates the control input sequences, only the first element of the sequence is applied to the system. In the next time step, the optimization problem is repeated over the shifted horizon to find a new sequence that may differ from the previous one. MPC computes the optimization in each sample rather than holding the control signal constant over the prediction horizon due to the fact that there could be a deviation between prediction and the system behavior through each step. This receding computation enables the regulation of the deviation from one sample to another. Solving an optimization algorithm at each sample time step is costly according to sample time and memory usage limitations. However, the computing control variable at each step has the advantage of rendering the system more responsive to disturbances and more robust to modeling errors or uncertainties on the model or its parameters.

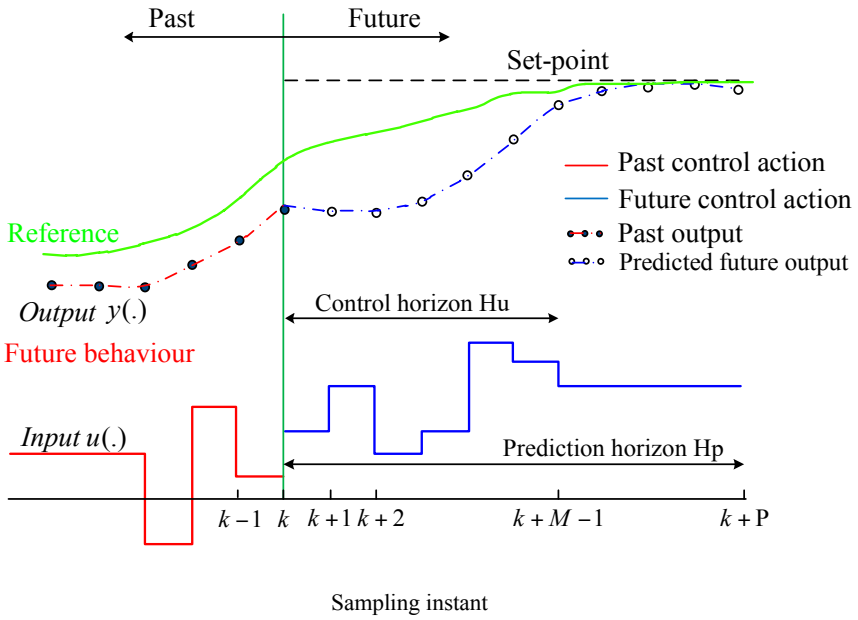

Figure 2.1: The illustration of the Receding Horizon Control idea. In each step, the horizon is shifting over time to generate the optimal control sequence.

2.2 Problem Formulations

2.2.1 System Model

Model predictive controllers rely on parametric dynamic models of the process. These models can be obtained via empirical techniques, e.g., linear data-driven system identification, via modeling using first principles, e.g., modeling mechanical and electrical components, or a mixture of both approaches. MPC chooses the best control action based on predictions by solving the optimal control problem over a pre-defined horizon. MPC uses a system model as the algorithm's fundamental property to predict the future responses of the system. The linear time-invariant(LTI) system model is considered in the form of discrete state-space form in this thesis, and it is given by the following equations,

$$x_{k+1} = Ax_k + B_u u_k \tag{2.1}$$

$$y_k = Cx_k \tag{2.2}$$

Here $x_k \in \mathbb{R}^{n_x}$ represents the states of the system, $u_k \in \mathbb{R}^{n_u}$ is control signal, and $y_k \in \mathbb{R}^{n_y}$ is the output vector. The A, B_u and C matrices are used to describe the system.

Sensing elements play a crucial role in feedback control strategies. Most of the practical applications include related sensing elements to measure the system states. Even if all states in the system are measurable in most cases, they may be noisy in practical application due to uncertainty in measurement or switching noise in the driver. MPC utilizes an observer by using the discrete state-space form of the system model instead of direct usage of measurements to increase the quality of the state information and reduce the effects of measurement noises, which lead to undesirable ripples in the steady-state. In this context, MPC uses the output of the observer at each step in the computation of control actions, which is very a common practice in practical MPC applications[14, 15]. Thanks to its simple and effective structure, we implement the well-known observer method, i.e., Luenberger Observer,

$$\hat{x}_{k+1} = A\hat{x}_k + Bu_k + L(y_k - C\hat{x}_k) \tag{2.3}$$

where $\hat{x}_k$ is the estimated state. L is user-specified observer matrix and it is selected as steady-state Kalman filter gain.

The noticeable absent topic of this thesis is to provide background for the stability of MPC. However, the primary focus of this work is not to investigate the stability of the closed loop system as it is proved with several approaches over the past three decades [16, 7, 17, 18]. The literature survey on the stability investigation of MPC in [16] formalized the conditions for closed loop stability for different cases, e.g., open loop stable/unstable. It is shown that the nominal stability is achieved via extending the prediction horizon that captures the dynamics of open loop stable or introducing terminal set or terminal constraint that drive the states to predefined set at the end.

2.2.2 Augmented State Formulations

It is inevitable in the practical applications that the model includes uncertainties or parameter mismatches that lead to undesirable effects on the system's performance or causes to have offset for reference tracking applications. MPC for LTI systems assumes that the system model or its parameters are constant over the prediction. Therefore, the algorithm requires some external control mechanism to handle this issue. On the other hand, there could be some other modifications to the system state based on the system requirements. In this section, we introduce new states or modifications to the states that are presented to fulfill system requirements.

2.2.2.1 Offset-free tracking

In order to have zero offsets in tracking the desired set-point, integral action, which is the well-known method in the feedback control, is presented in the algorithm. It is also possible to avoid the offset in the predictions due to modeling error by using this approach. A common technique to utilize the integral action is introducing the

integrator state by extending the existing state-space model.

$$\begin{bmatrix} x_{k+1} \\ x^i_{k+1} \end{bmatrix} = \begin{bmatrix} A & 0 \\ -C & I \end{bmatrix} x_k + \begin{bmatrix} B \\ 0 \end{bmatrix} u_k + \begin{bmatrix} 0 \\ I \end{bmatrix} r_k \tag{2.4}$$

$$y_k = \begin{bmatrix} C & 0 \end{bmatrix} x_k \tag{2.5}$$

2.2.2.2 Alternative State Choices

In the standard formulation of the LTI system, u is taken as the input while x represents the system states. However, it is sometimes more convenient to express the system model with the combination or modified form of the existing states or inputs. One might as well desire to penalize the rate of change of the system input. Therefore, the algorithm requires a new augmented state that represents the change of the input. Furthermore, the change of the control signal goes to zero as tracking performance is satisfied, i.e., steady-state.

Introducing the new augmented state that meets the specification as,

$$\Delta u_k \overset{\text{def}}{=} u_k - u_{k-1} \tag{2.6}$$

$$u_k = u_{k-1} + \Delta u_k \tag{2.7}$$

$$x_{k+1} = Ax_k + Bu_k = Ax_k + Bu_{k-1} + B\Delta u \tag{2.8}$$

$$\text{Define: } \tilde{x}_k = u_{k-1} \tag{2.9}$$

$$\begin{bmatrix} x_{k+1} \\ \tilde{x}_{k+1} \end{bmatrix} = \begin{bmatrix} A & B \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ \tilde{x}_k \end{bmatrix} + \begin{bmatrix} B \\ 1 \end{bmatrix} \Delta u_k \tag{2.10}$$

The boundary is related to integrating the produced optimization variable, i.e., change of the input. Fig. 2.2 represents this relation that includes the discrete integral in the intersection of the plant and the controller.

2.2.3 Generating State Matrices

The fundamental property of MPC is to predict the system to generate the best possible selections of the control input sequence. The number of the predicted future steps is determined by the H_p parameter. As is mentioned, the algorithm has the ability to

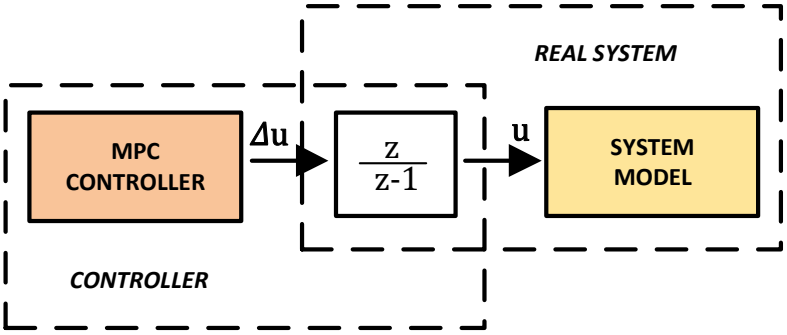

Figure 2.2: The illustration of the relation between MPC controller and the system itself. MPC produces the change of the control input, whereas the input is applied to the system after integration.

determine the number of H_u free moves of the manipulated input in the future. Since these two and other parameters are specified in the building of the algorithm step, future steps of the model are calculated offline and saved as constant values to feed the algorithm in the online calculations.

The number of the free moves in the future in terms of change of the control signals,

$$\begin{aligned} u_k &= u_{k-1} + \Delta u_k \\ u_{k+1} &= u_{k-1} + \Delta u_k + \Delta u_{k+1} \\ &\vdots \\ u_{k+H_u-1} &= u_{k-1} + \sum_{i=0}^{H_u-1} \Delta u_{k+i} \end{aligned} \tag{2.11}$$

and the system state x_k is predicted H_p steps in the future as,

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k = Ax_k + B(u_{k-1} + \Delta u_k) \\ x_{k+2} &= Ax_{k+1} + Bu_{k+1} = Ax_{k+1} + B(u_{k-1} + \Delta u_k + \Delta u_{k+1}) \\ x_{k+2} &= A^2 x_k + (B + AB)(u_{k-1}) + (B + AB)\Delta u_k + B\Delta u_{k+1} \\ &\vdots \\ x_{k+H_p} &= A^{H_p} x_k + \sum_{i=0}^{H_p-1} A^i B u_{k-1} + \sum_{i=0}^{H_p-1} A^i B \Delta u_k + \ \ldots \ \sum_{i=0}^{H_p-H_u} A^i B \Delta u_{k+H_u-1} \end{aligned} \tag{2.12}$$

Substituting the equation 2.12 in the equation in 2.1, the future output responses calculated as,

$$y_{k+1} = Cx_{k+1} = C(Ax_k + B(u_{k-1} + \Delta u_k)) \tag{2.13}$$

$$y_{k+2} = CA^2x_k + C(B + AB)(u_{k-1}) + C(B + AB)\Delta u_k + CB\Delta u_{k+1}$$

$$\vdots$$

$$y_{k+H_p} = C\left(A^{H_p}x_k + \sum_{i=0}^{H_p-1} A^iBu_{k-1} + \sum_{i=0}^{H_p-1} A^iB\Delta u_k + \ \ldots \ \sum_{i=0}^{H_p-H_u} A^iB\Delta u_{k+H_u-1}\right) \tag{2.14}$$

Together with, it can be defined as more compact form with the following equation for $(k+i)^{th}$ time instant,

$$y_{k+i} = CA^{k+i}x_k + C\sum_{w=0}^{i} A^wB\left(u_{k-1} + \sum_{j=k}^{w+k} \Delta u_j\right). \tag{2.15}$$

The output is in vector form with the size of $H_p \times 1$ and its equation stated as the following equation,

$$\begin{bmatrix} y_{k+1} \\ y_{k+2} \\ \vdots \\ y_{k+H_p} \end{bmatrix} = \Psi x_k + \Omega u_{k-1} + \Theta \begin{bmatrix} \Delta u_k \\ \Delta u_{k+1} \\ \vdots \\ \Delta u_{k+H_u-1} \end{bmatrix} \tag{2.16}$$

where

$$\Theta = \begin{bmatrix} CB & 0 & \ldots & 0 \\ C(AB+B) & CB & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C(\sum_{i=0}^{H_p-1} A^iB) & C(\sum_{i=0}^{H_p-2} A^iB) & \ldots & C(\sum_{i=0}^{H_p-H_u} A^iB) \end{bmatrix} \tag{2.17}$$

$$\Omega = \begin{bmatrix} CB \\ C(AB+B) \\ \vdots \\ C(\sum_{i=0}^{H_p-1} A^iB) \end{bmatrix}, \quad \Psi = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^{H_p} \end{bmatrix}$$

2.2.4 Casting the Formulations into Quadratic Program(QP) Problem

The core of the optimal control problem is introducing the cost function to generate the optimal control moves based on the requirements. In almost all feedback control systems, the primary goal of the control algorithm is to track the desired set-point. The cost function also includes other elements in order to have the desired characteristic. As an example, consider the cruise control in the vehicles. The main objective is to follow the predetermined desired speed from the vehicle. In addition, one might include the energy efficiency in the cost function that enables one to consider the fuel quantity. For this case, the control algorithm has to ensure the balance between tracking and the control effort, i.e., energy efficiency, so that it produces optimal control input in each step. The balance, i.e., the penalizing of each element in the cost function based on their significance, is determined by the weighting gain matrices.

There are several ways to define the related cost function, e.g., l_1 norm, l_2 norm. It is common to utilize the l_2 norm in order to make the cost function quadratic. The cost function fundamentally determines the desired performances of the system, i.e., minimizing a weighted sum of tracking error and control effort, which is dominantly formulated as a quadratic cost function. In addition to reducing the quadratic cost function to achieve the desired level of control performance, MPC must also ensure that state, output, and input trajectories satisfy some constraints, generally formulated as linear equalities and inequalities. In that respect, the sub-problem of MPC at each step turns into a quadratic programming problem. In this context, the quadratic programming problem of the MPC action takes the following form

$$\min_{\Delta u} \sum_{i=1}^{H_p} ||Q^{\frac{1}{2}}(y_{k+i} - r_k)||_2^2 + \sum_{j=0}^{H_u-1} ||R^{\frac{1}{2}}\Delta u_{k+j}||_2^2 \tag{2.18a}$$

$$\text{subject to: } x_{k+i+1} = Ax_{k+i} + B_u u_{k+i} \tag{2.18b}$$

$$y_{k+i+1} = Cx_{k+i+1} \tag{2.18c}$$

$$u_{k+i} = u_{k+i-1} + \Delta u_{k+i} \tag{2.18d}$$

$$u_i^{min} \leq u_{k+i} \leq u_i^{max} \tag{2.18e}$$

$$y_i^{min} \leq y_{k+i+1} \leq y_i^{max} \tag{2.18f}$$

$$\Delta u_{k+j+N_u} = 0 \tag{2.18g}$$

$$j = 0, \ldots, H_p - H_u - 1, i = 0, \ldots, H_p - 1 \tag{2.18h}$$

where Q and R are the positive definite weight matrices associated with tracking error and input effort, respectively. Feasible values are selected for H_p and H_u so as to keep the balance between capturing the dynamics of the system and reducing the complexity of the resulted QP problem[19].

Let us rewrite the cost function in (2.18) in a more compact form to cast it in quadratic form. We denote $\mathcal{J}$ to represent the value of the cost function,

$$\mathcal{J} = (y_k - y_k^{ref})^T Q (y_k - y_k^{ref}) + \Delta u_k^T R \Delta u_k \tag{2.19}$$

The weight matrices shall be positive definitive or semi-definite for the convexity of the problem. It is also worth noticing that one may adjust each element in the gain matrices for their purposes, and off-diagonal elements are usually left as zero if there are no alternative formulations.

To collect the cost function with respect to Δu, i.e., optimization variable, equation in (2.16) is substituted into (2.19)

$$\begin{aligned} \mathcal{J} = \Delta u_k^T \underbrace{(\Theta^T Q \Theta)}_{\mathrm{H}} \Delta u_k - \underbrace{2\Theta^T Q (y_k^{ref} - \Psi x_k - \Omega u_{k-1})}_{g_k} \Delta u + \ldots \\ \cdots + \underbrace{(y_k^{ref} - \Psi x_k - \Omega u_{k-1})^T Q (y_k^{ref} - \Psi x_k - \Omega u_{k-1})}_{\text{constant}} \end{aligned} \tag{2.20}$$

Since the constant(bias) term has no effect on the decision variable but the value of the cost function, it can be excluded from the cost function. The resulting terms depend only on the optimization variable. As we have that H term is positive definite, it is clear that the result of the optimization problem will be the global optimum.

In order to include the constraints into the quadratic problem, three polyhedral sets are defined to cover the change of input, input, and states.

The first constraint is on the change of the input, which is also the optimization variable. It is simply defined as putting the bounds on the future predicted value of the

decision variable in the form of a linear constraint.

$$\begin{bmatrix} -I & 0 & \dots & 0 \\ 0 & -I & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & -I \\ I & 0 & \dots & 0 \\ 0 & I & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & I \end{bmatrix} \begin{bmatrix} \Delta u_k \\ \Delta u_{k+1} \\ \vdots \\ \Delta u_{k+H_u-1} \end{bmatrix} \leq \begin{bmatrix} -\Delta u_{min} \\ -\Delta u_{min} \\ \vdots \\ -\Delta u_{min} \\ \Delta u_{max} \\ \Delta u_{max} \\ \vdots \\ \Delta u_{max} \end{bmatrix} \tag{2.21}$$

Rather than formulating the input increment, the second type constraint is formulated by introducing integration matrices of the decision variable to cover the manipulated input itself. The limits on the voltage or the valve position can be examples of the input constraint.

$$\begin{aligned} u_k &= u_{k-1} + \Delta u_k \\ u_{k+1} &= u_{k-1} + \Delta u_k + \Delta u_{k+1} \\ &\vdots \\ u_{k+H_u-1} &= u_{k-1} + \Delta u_k + \cdots + \Delta u_{k+H_u-1} \end{aligned} \tag{2.22}$$

or

$$\begin{bmatrix} u_k \\ u_{k+1} \\ \vdots \\ u_{k+H_u-1} \end{bmatrix} = \begin{bmatrix} I \\ I \\ \vdots \\ I \end{bmatrix} u_{k-1} + \begin{bmatrix} I & 0 & \dots & 0 \\ I & I & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ I & I & \dots & I \end{bmatrix} \begin{bmatrix} \Delta u_k \\ \Delta u_{k+1} \\ \vdots \\ \Delta u_{k+H_u-1} \end{bmatrix} \tag{2.23}$$

or augmented together:

$$\begin{bmatrix} -I & 0 & \dots & 0 \\ 0 & -I & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & -I \\ I & 0 & \dots & 0 \\ 0 & I & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & I \end{bmatrix} \begin{bmatrix} \Delta u_k \\ \Delta u_{k+1} \\ \vdots \\ \Delta u_{k+H_u-1} \end{bmatrix} \leq \begin{bmatrix} -u_{min} + u_{k-1} \\ -u_{min} + u_{k-1} \\ \vdots \\ -u_{min} + u_{k-1} \\ u_{max} - u_{k-1} \\ u_{max} - u_{k-1} \\ \vdots \\ u_{max} - u_{k-1} \end{bmatrix} \tag{2.24}$$

The last constraint is defined to bound the states in the system. There could be some constraints that require special attention, such as avoidance of forbidden zones or currents not to exceed its rated value in the electric motor. The formulation of constraint on states takes the system model into account to predict the future responses whether they violate the limits in the present or future.

By using the equation in (2.16), the constraints on the output of the system with respect to Δu is given via the following equation:

$$\begin{bmatrix} -\Theta \\ \Theta \end{bmatrix} \begin{bmatrix} \Delta u_k \\ \Delta u_{k+1} \\ \vdots \\ \Delta u_{k+H_u} \end{bmatrix} \leq \begin{bmatrix} -\begin{bmatrix} y_{min} \\ y_{min} \\ \vdots \\ y_{min} \end{bmatrix} + \Psi x_k + \Omega u_{k-1} \\ \begin{bmatrix} y_{max} \\ y_{max} \\ \vdots \\ y_{max} \end{bmatrix} - \Psi x_k - \Omega u_{k-1} \end{bmatrix} \tag{2.25}$$

Given system structure and constraints formulation, the quadratic optimization problem can be defined as:

$$\min_z \ \frac{1}{2} z^T H z + z^T g \tag{2.26a}$$

subject to

$$Wz \leq b \tag{2.26b}$$

Algorithm 1 presents the summary of the MPC updating process in each sample time. Most of the algorithm inputs can be calculated offline that saves computational time. The timing diagram in Fig. 2.3 illustrates the online calculations in each sample step

Algorithm 1 MPC updating procedure in each sample step

Input: (H, g, W, b)

1 : $y_k \leftarrow$ collect measurement from the system:

2 : Compute error based on the present reference r_k and output y_k

3 : $(g, b) \leftarrow$ Update g and b vectors corresponding linear terms in the quadratic problem 2.26

4 : $z_k \leftarrow$ solve the optimization problem such that:

$\min_z \ \mathcal{J}(z_k) : \frac{1}{2} z_k^T H z_k + z_k^T g$

subject to: $W z_k \leq b$

5 : $\Delta u_k \leftarrow$ select the first element from the H_u optimum sequence

6 : $u_k \leftarrow u_{k-1} + \Delta u_k$ integrate to calculate the optimal control inputs

7 : $\hat{x}_{k+1} \leftarrow$ Update states via observer

$\hat{x}_{k+1} = A\hat{x}_k + Bu_k + L(y_k - C\hat{x}_k)$

to compute new control actions for the system.

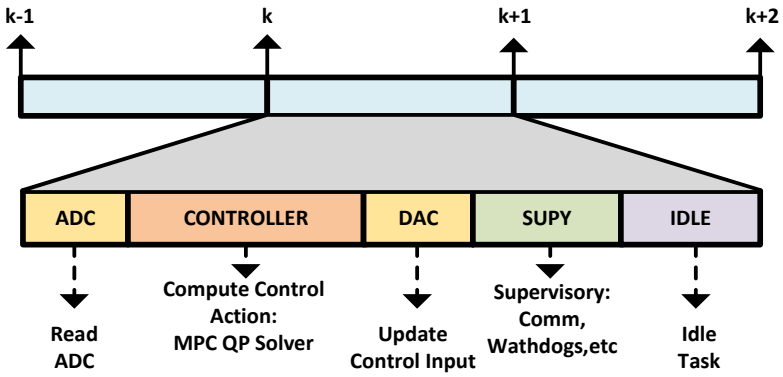

Figure 2.3: The overall timing diagram to execute the online calculations in each sample step.

CHAPTER 3

ONLINE OPTIMIZATION IN MPC

Solving the optimization problem in each sample plays a crucial role in the constrained MPC. As is mentioned, it is possible to transform MPC formulation into a convex quadratic optimization problem with linear constraints for linear-dynamical systems. Thus, researchers and engineers have long been focused on adapting quadratic programming(QP) techniques in the MPC domain. One of the critical tasks in online MPC is to compute the solution of the problem 2.26 to give an "accurate" solution within the sampling time, which is generally in the order of milliseconds or even sub milliseconds for the majority of the modern control applications. In this thesis, one of our main focuses is to efficiently implement an active set method that provides the solution inside the allocated time for QP and uses minimal memory space. Before starting to introduce the solving methods for QPs, let us define the parametric QP,

Definition 3.0.1 *The QP is called as parametric QP, if it defined as,*

$$\min_{z} \ \frac{1}{2} z^T H z + z^T g_k \tag{3.1a}$$

$$\textit{s.t. } W z \leq b_k \tag{3.1b}$$

where g_k and b_k are dynamically changing vectors depending on the information of the system at k^{th} step.

The reason to define the parametric QP is that only the constraint and gradient vectors in the MPC formulation are changing over each sample step based on the optimization variable. The Hessian and the constraint matrix, i.e., left half of the constraint relation, are calculated offline and saved as constant values to be used throughout the operations. This section of the thesis aims to provide the methods for solving the

QP by starting from an unconstraint case. As a further contribution, hints and some suggestions for the practical implementation are also addressed.

3.1 Unconstrained Case in QP

The solution of the unconstrained QP is straightforward as opposed to the constraint case since the Hessian matrix is constructed to be as positive definite, so the problem is strictly convex. Therefore, it only checks the first-order condition to find the optimum global point, i.e.,

$$\nabla(\mathcal{J}) = 0 \text{ that is, } Hz + g_k = 0 \tag{3.2}$$

Since we deal with finding the solution for parametric QPs, i.e., gradient g is the only parameter that changes over each step, and it is trivial to obtain the solution via only matrix-vector multiplication.

$$\Delta u = -H^{-1} g_k \tag{3.3}$$

On the other hand, there are several approaches in the literature[4] to find the optimal solution for unconstrained QPs where the inverse of the Hessian matrix is required in each sample. For those cases, taking the inverse of the matrix costs heavily in terms of computational complexity. Therefore, one technique is to apply Cholesky decomposition to calculate the inverse of the matrix to reduce the computational burden. In this approach, H matrix is assumed as positive definite. It can be decomposed as the product of the two matrices, i.e., multiplication of lower triangular and its conjugate transpose.

$$H = LL^T \tag{3.4}$$

The inverse of the matrix is then calculated by applying forward and back substitution methods, respectively.

$$L \underbrace{L^T z}_{y} = -g_k \xrightarrow[\text{back-subs}]{} Ly = -g_k \xrightarrow[\text{forward-subs}]{} y^* \tag{3.5}$$

3.2 Active-Set Methods for Solving QP

The idea of the active set method is to impose all constraints as equalities when they are active at the current iteration. In other words, this method uses the fact that the solution of the constraint QP lies on the boundary of the constraints. Otherwise, the solution will be inside the constraints, which is the optimum global point, i.e., unconstrained case. The problem is reduced to the sub-problem of QP with equality constraints i.e. $W_{\mathbb{A}}z = b_{\mathbb{A}}$ where $\mathbb{A}$ denotes active set.

$$\min_{z} \ \frac{1}{2}z^T H z + z^T g_k \tag{3.6a}$$

subject to

$$W_{\mathbb{A}}z = b_{\mathbb{A}} \tag{3.6b}$$

The solution is found iteratively starting from a feasible point by adding the violated constraint into or dropping the blocking constraint from the active set in each iteration. The active set method has different variations: *primal*, *dual*, and *primal-dual*[5].

The primal method starts from the primal feasible initial point z^0 and iterates by maintaining primal feasibility in each iteration until dual feasibility (i.e., Lagrange multipliers is greater than zero) is reached[5, 20]. The algorithm requires an auxiliary method, e.g., the simplex method, to provide the initial feasible point. It is also possible to obtain the initial feasible working set when the initial feasible point is found. Note that the number of constraints might be more than the size of the optimization variable(n). The range space of the solution space can be represented by the linear combination of n linearly independent vectors. Therefore, the working set includes linearly independent constraint vectors only up to n, since the other constraints are only the linear combination of the constraints inside the working set.

The next iteration of the optimization variable is chosen as,

$$z^{k+1} = z^k + \tau^k \Delta z^k, \ \tau \in \mathbb{R}_{\geq 0} \tag{3.7}$$

The Δz represents the direction point from the present point to the optimal point. Once the Δz equals zero, i.e., no more direction, the optimality conditions(Lagrange multipliers) must be checked. If there are negative Lagrange multipliers, then the

corresponding constraint must be dropped from the active set since it is no longer a feasible constraint for the present point. If there are more negative Lagrange multipliers than one, then the constraint related to the most negative multiplier must be dropped.

On the other hand, if the Δz is different from zero, the algorithm takes a step in the direction of Δz with step length τ. The reason to include the τ term as the product of the Δz is to avoid the primal infeasibility since there can be a violation in the non-active constraints when moving from z^k to z^{k+1}. If there is a violation, the step size of the direction is determined to satisfy the corresponding violation. The related constraint is called a blocking constraint and must be added to the active set. The value of the step length is determined as follows,

$$\tau = \min\Big(1, \min_{i \notin W_k} \big(\frac{W_q^T z^i - b_q}{W_q^T \Delta z^i} \mid W_q^T \Delta z^i < 0\big)\Big) \tag{3.8}$$

The active-set method adds or drops only a constraint in each iteration. Thus, the algorithm shows its strength for medium or small-size problems, which is the case of most embedded MPC in terms of execution speed and solution accuracy[21, 22]. Furthermore, the algorithm reaches the optimum point, even in ill-conditioned problems, in a finite number of iteration[23]. We provide the summary of the primal method active-set method via Algorithm 2.

Algorithm 2 Primal Active Set Method

Input: Generate a (non-optimum) feasible starting point z_0 which satisfy $Wz \leq b$. Create the initial working set with constraints active $W_0 = \{z | W_i^T z_k = b_i\}$

for k=0... **do**

Construct the active set matrix $W_{\mathbb{A}} = [a_i^T]$, $i \in W_k$ and solve KKT(Karush-Kuhn-Tucker):

$$\begin{bmatrix} H & W_{\mathbb{A}}^T \\ W_{\mathbb{A}} & 0 \end{bmatrix} \begin{bmatrix} \Delta z^* \\ \lambda^* \end{bmatrix} = \begin{bmatrix} -(Hz_k + g_k) \\ 0 \end{bmatrix} \tag{3.9}$$

if $\Delta z == 0$ **then**

Check all Lagrange multipliers λ_i corresponding constraints in the Active Set

if $\lambda_i \geq 0$ **then stop** and **return** optimal point: i.e. (z^*)

else

Remove the constraint, which has the most negative λ from W_k

end if

else

Compute the step length via Eq-3.8

Update the optimization variable via Eq-3.7

if Blocking constraint found **then**

Add it to the current active set,i.e., $\mathbb{A}^{i+1} \leftarrow \mathbb{A}^i \cup q$

end if

end if

end for

3.2.1 Dual Active Set Solver

This section describes the details of the dual active set solver, which is a very attractive method for solving small and medium-size problems and provides infeasibility if there is no solution for related QP. As opposed to the primal method, the dual method starts from dual feasible initial points z^0,λ^0. It aims for primal feasibility and maintains dual feasibility in each iteration until no violated constraints exist[24, 25]. In this work, we use a dual-active set solver that is based on the works of Goldfarb and Idnani [24]. The pair (z^*,λ^*) is the optimal point of the problem (2.26) with equality constraints if they satisfy the first-order necessary condition by solving the following linear equation:

$$\underbrace{\begin{bmatrix} H & W_{\mathbb{A}}^T \\ W_{\mathbb{A}} & 0 \end{bmatrix}}_{\text{KKT}} \begin{bmatrix} z \\ \lambda \end{bmatrix} = \begin{bmatrix} -g_k \\ b_k \end{bmatrix} \tag{3.10}$$

The solution of (3.10) exists if the KKT matrix is not singular. MPC formulation enables that the Hessian matrix H is positive definite i.e. nonsingular, and if $W_{\mathbb{A}}$ is full row-rank in the current iteration, the inverse of the matrix exists.

Several direct methods can be applied to explicitly take inversion of the KKT matrix, such as Schur complement, null-space, and range-space approach[20]. The range-

space approach is used to express explicit inversion of the KKT matrix.

$$\begin{bmatrix} H & W_{\mathbb{A}}^T \\ W_{\mathbb{A}} & 0 \end{bmatrix} \begin{bmatrix} H^{-1}(I - W_{\mathbb{A}}^T \mathcal{Z} W_{\mathbb{A}} H^{-1}) & H^{-1} W_{\mathbb{A}}^T \mathcal{Z} \\ \mathcal{Z} W_{\mathbb{A}} H^{-1} & -\mathcal{Z} \end{bmatrix} = I \tag{3.11}$$

where $\mathcal{Z} = (W_{\mathbb{A}} H^{-1} W_{\mathbb{A}}^T)^{-1}$. To calculate the pair (Δz,$\Delta\lambda$), two operators are defined as;

$$\mathcal{K} = \mathcal{Z} W_{\mathbb{A}} H^{-1} \tag{3.12}$$

$$\mathcal{G} = H^{-1}(I - W_{\mathbb{A}}^T \mathcal{K}) \tag{3.13}$$

where $W_{\mathbb{A}} \in \mathbb{R}^{n_a x n}$.

Goldfarb and Idnani [24] suggest that the dual-method can use the unconstrained optimum point of the objective function as their initial value.

$$(z^0, \lambda^0) \leftarrow (-H^{-1} g, 0) \tag{3.14}$$

In contrast, the primal method needs an auxiliary algorithm that provides a feasible starting point and active set. Thus, the dual-method, except for some basic matrix operations, does not require a preliminary calculation phase as in the primal method. Also, the dual-method finds the optimal solution with fewer iterations compared to the primal method that makes the dual method more efficient than primal method [24]. In addition, the solution of the unconstrained problem is trivial in the dual-method since it starts with the unconstraint optimum point.

The primal and dual variables are updated in each iteration according to the equation (3.15) so that the most violated constraint becomes active(feasible) at iteration k while maintaining dual feasibility.

$$\begin{aligned} z^{i+1} &= z^i + \alpha^i \Delta z^i, \\ \lambda^{i+1} &= \begin{bmatrix} \lambda^i \\ \lambda_p^k \end{bmatrix} + \alpha^i \begin{bmatrix} \Delta\lambda^i \\ 1 \end{bmatrix} \end{aligned} \tag{3.15}$$

where the direction variables Δz, $\Delta\lambda$ are updated via the following equations,

$$\Delta z^i = \mathcal{G} W_{\mathbb{A}}^T, \tag{3.16}$$

$$\Delta\lambda^i = \mathcal{K} W_{\mathbb{A}}^T \tag{3.17}$$

In opposition to the primal method, there are two-step length parameters in the dual method for primal and dual variables, respectively. The step length must be minimum enough to satisfy the violated constraint, if exists, and also must be minimum enough to maintain the dual feasibility. Therefore, primal and dual step lengths are firstly determined separately to fulfill their properties. As a result, their minimum value is taken to be used in the algorithm to maintain both feasibilities.

Three cases occur according to the value of the primal-dual step length. It is either equals to the primal or dual variable if their value is less than ∞. The dual method can detect whether there is an infeasibility by using the primal-dual step length, i.e., if the value is ∞, there is no direction to take.

1. Full step can be taken if $\alpha^i = \alpha^i_{primal}$
2. Partial step can be taken if $\alpha^i = \alpha^i_{dual}$ and the k^{th} blocking constraint must be dropped from active set such that $k \leftarrow \underset{k \in \mathbb{A}}{\operatorname{argmin}}(-\frac{\lambda^i}{\Delta\lambda^i} | \Delta\lambda^i_k < 0)$
3. Problem is infeasible if $\alpha^i = \infty$

Note that if the algorithm terminates with infeasibility result, the optimization variable Δu can be set to either its previous or to zero since its terminated value might be undesirable to the system. Algorithm 3 summarizes the processes in the dual active set solver.

Algorithm 3 Dual Active Set Solver[24]

Input: (H, g, W, b)

Step-1. Initialization of the algorithm:

set $i \leftarrow 0$, $(z, \lambda) \leftarrow (-H^{-1}g, 0)$, $\mathbb{A} \leftarrow \emptyset$, $J \leftarrow Z^{-1}$, $n^0_a \leftarrow 0$

Step-2. Constraint violation check:

set $q \leftarrow \begin{cases} k \in (\mathbb{I} \backslash \mathbb{A} | (W_k z^i_k - b_k) > 0) \\ 0, \quad \text{otherwise} \end{cases}$

Step-3. Termination condition check:

if $q = 0$ **then**, **stop** and **return** optimal active set: i.e. (z^*, λ^*) and corresponding active set $\mathbb{A}^i(z^*)$

end if

Step-4. Calculate step directions:

$\Delta z^i = \mathcal{G} W_{\mathbb{A}}^T$ (primal direction)

if $n_a^i > 0$ **then**, calculate the dual direction

$\Delta \lambda^i = \mathcal{K} W_{\mathbb{A}}^T$ (dual direction)

end if

Step-5. Calculate step lengths:

$$\alpha_{primal}^i \leftarrow \begin{cases} \infty, & \text{if } \|\Delta z^i\| == 0 \\ \frac{W_q^T z^i - b_q}{W_q^T \Delta z^i}, & \text{otherwise} \end{cases}$$

$$\alpha_{dual}^i \leftarrow \begin{cases} \infty, & \text{if } \nexists k \\ -\frac{\lambda_k^i}{\Delta \lambda_k^i}, & \text{otherwise} \end{cases} \quad \text{s.t. } \underset{k \in \mathbb{A}}{\operatorname{argmin}}(-\tfrac{\lambda^i}{\Delta \lambda^i} | \Delta \lambda_k^i < 0)$$

$\alpha^i \leftarrow min(\alpha_{primal}^i, \alpha_{dual}^i)$

Step-6. Check for feasibility and take a partial step in dual space:

if $\alpha_{primal}^i = \infty$ **then**

  if $\alpha_{dual}^i = \infty$ **then**,stop and **return** QP is infeasible

  else

    Drop blocking constraint i.e.

    $\mathbb{A}^{i+1} \leftarrow \mathbb{A}^i \backslash k$: k calculated in **Step-5**

    $\lambda_p^{i+1} \leftarrow \lambda_p^i + \alpha^i$, $\lambda^{i+1} \leftarrow \lambda^i + \alpha^i \Delta \lambda^i$

    $n_a^{i+1} \leftarrow n_a^i - 1$, and $i \leftarrow i + 1$

    Update R and J matrices and go to **Step-4**

  end if

end if

Step-7. Take a full step in primal and dual space

$z^{i+1} \leftarrow z^i + \alpha^i \Delta z^i$, $\lambda^{i+1} \leftarrow \lambda^i + \alpha^i \Delta \lambda^i$, $\lambda_p^{i+1} \leftarrow \lambda_p^i + \alpha^i$

if $\alpha^i = \alpha_{primal}^i$ **then**

  Add current violated constraint in the active set i.e. $\mathbb{A}^{i+1} \leftarrow \mathbb{A}^i \cup q$

  $n_a^{i+1} \leftarrow n_a^i + 1$, and $i \leftarrow i + 1$

  Update R and J matrices and go to **Step-2**

else

  Drop blocking constraint i.e.

  $\mathbb{A}^i \leftarrow \mathbb{A}^{i-1} \backslash k$: k calculated in **Step-5**

  $n_a^{i+1} \leftarrow n_a^i - 1$, and $i \leftarrow i + 1$

  Update R and J matrices and go to **Step-4**

end if

Output: (z^*, λ^*) and $\mathbb{A}(z^*)$ and return flag

We also draw the flowchart of the dual active set solver in order to make the algorithm plain and understandable. Fig. 3.1 illustrates the overall progress in the algorithm in terms of flow diagrams.

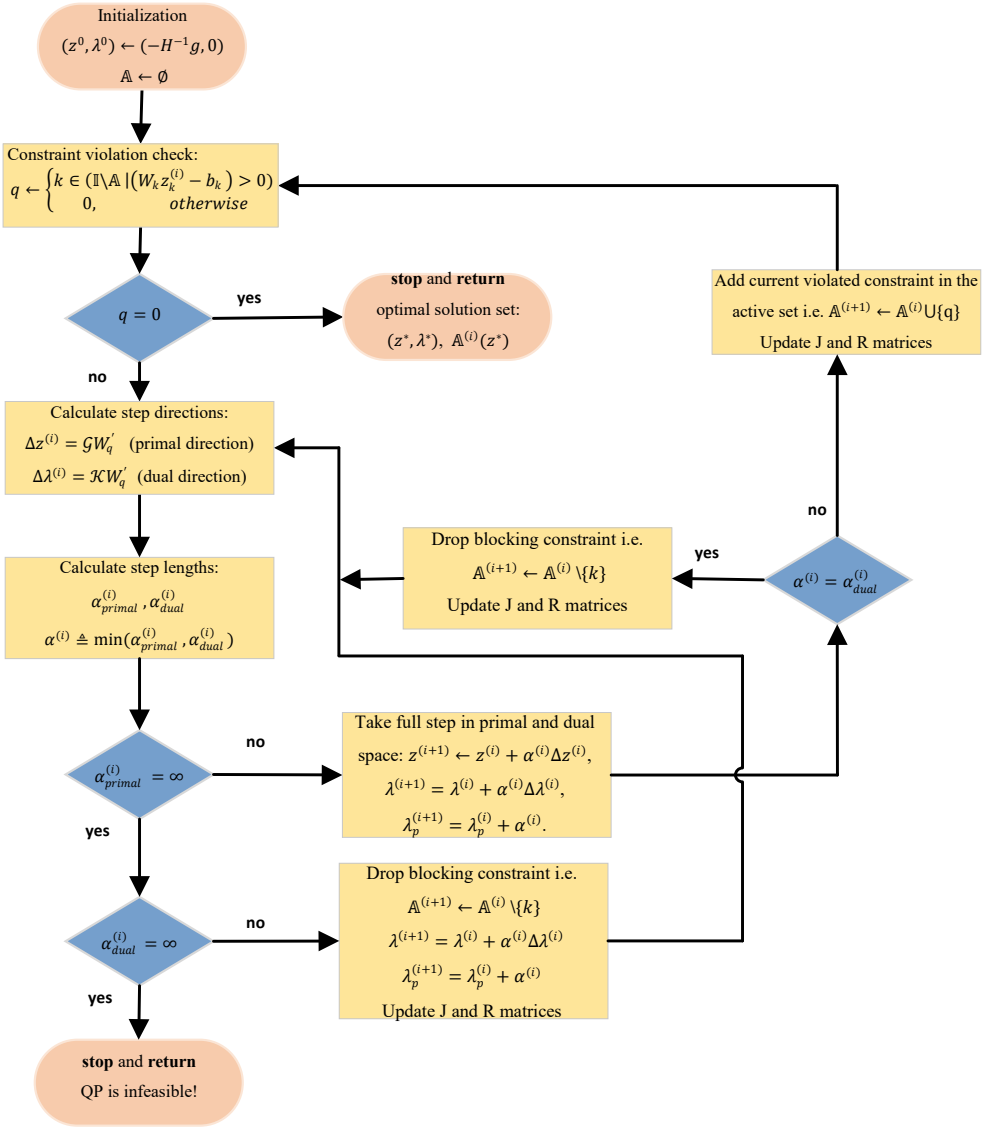

Figure 3.1: The illustration of a flowchart for the dual active set solver.

We utilize matrix factorizations to calculate primal- and dual-step directions in Step 4 instead of updating operators, $\mathcal{G}$ and $\mathcal{K}$, explicitly in each iteration for computational

efficiency. Let Cholesky decomposition of H be

$$H = ZZ^T \tag{3.18}$$

where Z is the lower triangular matrix, and QR decomposition of L:

$$L = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix} \tag{3.19}$$

such that

$$L = Z^{-1} W_{\mathbb{A}}^T. \tag{3.20}$$

Rewriting the operators in (3.12) and (3.13) by replacing Cholesky and QR decompositions, we can obtain

$$\begin{aligned} \mathcal{G} &= J_2 J_2^T \\ \mathcal{K} &= R^{-1} J_1 \end{aligned} \tag{3.21}$$

where

$$J = \begin{bmatrix} J_1 & J_2 \end{bmatrix} = L^{-T} \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \tag{3.22}$$

In the section 3.2.2, we provide the details of the efficient matrix updating procedure to accelerate the algorithm in terms of speed.

3.2.2 Implementation Details of Efficient Matrix Updating Strategy

Givens Rotation and Householder Reflection are two common approaches to compute the QR factorization[26]. The details of the two algorithms are described with a basic example in Appendix A for the ease of the reader. In the active set method, there is only single constraint addition to or deletion from the active set in each iteration. Therefore, it is more costly to utilize QR decomposition of L in each iteration than updating J and R matrices via numerically stable methods [24]. The algorithm starts with the $J = Z^{-T}$ and $R = \emptyset$ as their initial point and then, continuing to update them whenever $W_{\mathbb{A}}$ matrix changes until the algorithm gives the optimum points. Our implementation uses the Householder Reflection method to update J and R matrices in the constraint addition step. When the new constraint is added into active set the

relations in (3.19) and (3.19) becomes;

$$Z^{-1}\begin{bmatrix} W_{\mathbb{A}}^T & W_i^T \end{bmatrix} = Q\begin{bmatrix} R & r_{n_i^1} \\ 0_{(n-n_a)x(n_a-1)} & r_{n_i^2} \end{bmatrix} \tag{3.23}$$

There is only one rotation matrix applied to every new constraint to make the R matrix upper triangular due to they enter into the active set from the end. Applying the rotation matrix to each side;

$$Q_n Z^{-1}\begin{bmatrix} W_{\mathbb{A}}^T & W_i^T \end{bmatrix} = Q_n Q\begin{bmatrix} R & r_{n_i^1} \\ 0_{(n-n_a)x(n_a-1)} & r_{n_i^2} \end{bmatrix} \tag{3.24}$$

$$= \widehat{Q}\begin{bmatrix} R & r_{n_i^1} \\ 0_{(1)x(n_a-1)} & r \\ 0_{(n-n_a-1)x(n_a-1)} & 0_{(n-n_a-1)x(1)} \end{bmatrix} \tag{3.25}$$

Since the rotation matrix Q_n is orthogonal, it does not hinder to hold the property (3.18) after multiplication with Z^{-1}[14]. Our implementation uses the Givens Rotation method to update matrices in the constraint deletion from the active set. Unlike the constraint addition step, there might be more rotations in the deletion step depending on the constraint place since the deletion can be anywhere inside the active set [27]. Fig.3.2 illustrates the updating procedure for both Givens Rotation and Householder Reflection method in the first constraint is added into the active set case. The Householder Reflection method takes the vector as a whole to rotate, whereas the Givens method updates it as a piece of the vector with size 2.

We choose two different methods for updating the matrices to increase the algorithm's computational efficiency and reduce the execution time. We compare the two algorithms in both addition and deletion steps over their total floating-point operations (flops) inside the algorithms, enabling us to assess the overall computational cost in the updating process independent of the adopted hardware. The number of flops in our measurements also includes $\sqrt{\ }$ operation in addition to 4 basic mathematical operations i.e. $(\pm, *, \div)$. We divide the operations into 2 subgroups; $(\pm, *)$ and $(\div, \sqrt{\ })$ since the number of the cycles to execute the $\div$ and $\sqrt{\ }$ operations cost more than the $(\pm, *)$ in almost all types of computing units.

To be more precise, suppose that the active set is empty and let n be the dimension of solution space. In the worst-case scenario, when the first constraint is added into

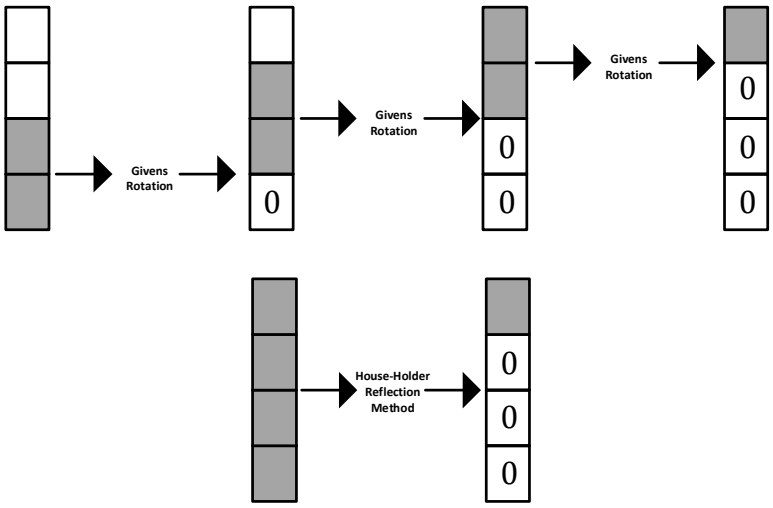

Figure 3.2: The illustration of the updating the vector with size 4 via Givens Rotation and Householder Reflection.

the active set, the methods shall perform $n-1$ calculations to transform R into upper triangular form if the Givens Rotation method is adopted. In contrast, there is a single column operation that enables less number of calculations in the Householder Reflection method. Suppose now that the active set has n constraints and the first column i.e., first-in constraint, is removed from the set. Both methods now shall perform $n-1$ step to complete updating the process. The number of flops needed to be executed in the Householder method is almost 50% less than the Givens Rotation in QR decomposition [28]. However, using the fact that the R is in upper Hessenberg form in constraint deletion brings more flexibility in executing fewer flops.

Table 3.1: The comparison of Householder and Givens Rotation Methods about their counts of operations

	House-Holder	Givens Rotation
Constraint Addition		
$(+,-,*)\|(\div,\sqrt{\ })$	$93\|2$	$84\|9$
Constraint Deletion		
$(+,-,*)\|(\div,\sqrt{\ })$	$147\|6$	$102\|9$

Table 3.1 provides the overall number of flops for constraint addition and deletion steps of both algorithms. The numbers in the table represent the total number in our efficient C-code implementation by taking the dimension of the solution space as 4. The count of $(\pm, *)$ is slightly different in the constraint addition, whereas there is a significant difference in the number of $(\div, \sqrt{\ })$ which is the main part of the execution speed. On the other hand, the main difference comes from $(\pm, *)$ operations in constraint deletion procedure that dominates the overall performance in terms of speed.

To be more precise, we provide an example according to the clock cycles information for operations in C28x processors provided by Texas Instruments. The $\div$ operation takes 24 clock cycles while $\sqrt{\ }$ operation takes 28 clock cycles [29]. On the other hand, 2 clock cycles are required to calculate $(+, -, *)$ operations [30]. Therefore, we can obtain the total clock cycles for both methods in constraint addition and deletion steps. The Householder Reflection method is more suitable in the constraint addition step since it only takes the 238 clock cycles to update the matrix as Givens Rotation requires 396 clock cycles. On the other hand, Givens Rotation shows its effectiveness in the constraint deletion step since 432 clock cycles are needed to update the matrix. In comparison, the Householder reflection method requires 450 clock cycles.

3.3 Real Time Implementation

This section covers the real-time MPC implementation and its comparison with literature examples to show the feasibility of the algorithm in terms of execution speed. The rotating antenna and Cessna Citation 500 examples are selected to analyze the efficiency of the proposed algorithm since there are several studies [31, 32, 33, 34, 1] in the literature to provide the execution speed of these two examples that enables us to compare them with our implementation. While we only provide the solution and comparison for the rotating antenna example, we examine some additional properties in Cessna Citation 500 example in terms of feasibility of the QP solver, e.g., scaling the original algorithm to eliminate the effect of the numerical problems. We verify the effectiveness of the algorithm via processors-in-the-loop(PIL) simulations in Matlab/Simulink environment by using the F28377S control card provided by Texas

Instruments.

3.3.1 Rotating Antenna Example

The goal of this problem is to regulate the angle difference between the antenna and the target so that the direction of the antenna always points at the target object [35]. The system, as illustrated in Fig. 3.3, basically consists of an antenna and electric motor, which is directly coupled. It is assumed that all state variables are measurable and defined as the position of the antenna (θ and $\theta_r rad$) and the angular velocity of the antenna ($\dot{\theta} rad/sec$). The CT state-space model is discretized via the forward-Euler method with sample time $T_s = 0.1s$ to obtain the discrete-time state-space model structure as

$$x_{k+1} = \begin{bmatrix} \theta_{k+1} \\ w_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0.1 \\ 0 & 0.99 \end{bmatrix} x_k + \begin{bmatrix} 0 \\ 0.0787 \end{bmatrix} u_k \tag{3.26}$$

$$y_k = \begin{bmatrix} 1 & 0 \end{bmatrix} x_k \tag{3.27}$$

For this example, MPC controller designed with H_p = 10 and H_u = 3 to make them comparable with the other applications. The tuning weights are specified as $R = 1$ and $Q = 3$. In addition, the voltage supplied to drive the electric motor is limited to

$$-2V \leq u_k \leq 2V \tag{3.28}$$

The system is firstly evaluated in the primal active set solver. We implement the primal method to analyze the efficiency of the second method. Fig. 3.4 illustrates the tracking performance, input voltage under constraint, and the execution time of the overall MPC algorithm with the primal method. The maximum execution time that includes the necessary MPC steps in addition to optimization solver is about $0.252\mu sec$. The desired target angle is applied as the step input with the change of $(0 - \pi)$ to operate the system as close to the constraint as possible for stressing the optimization solver. On the other hand, Figure 3.4 shows the result of efficient implementation of the dual active set method. We observe from the results that executing the algorithm via dual method approximately two times faster than primal method.

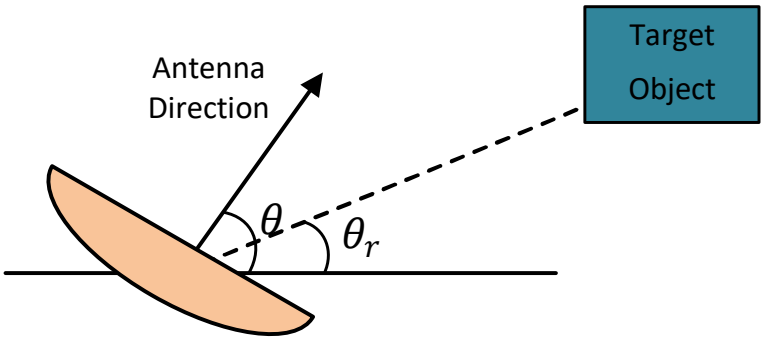

Figure 3.3: Basic diagram of antenna angular positioning system.

We compare the efficiency of the algorithm with other studies in the literature in terms of execution speed. Table-3.2 shows that the implementation of the efficient dual method has the best execution time performance among the other studies.

	Execution Times (ms) of the Rotating Antenna Example			
	Dual Active Set OP max time	Currie[31] Interior Point QP max time	Vouzis[1] Modified Newton max time[1]	Amira[32] Interior Point QP average time
Hp	10	10	10	4
Hc	3	3	3	3
$t_{sol}(ms)$	0.095	0.153	8.4	10.586

Table 3.2: Comparison execution time of the different MPC application for the rotating antenna example. [1] It is the estimated data based on the timing information in [1], that reported time is calculated by taking $H_p = 20$

3.3.2 Cessna Citation 500 Example

This example is provided by J.Maciejowski in [7] to validate the performances of the MPC under some set of constraints on both control input and states. The single-input multi-output(SIMO) state-space model is obtained by linearizing the dynamics of the Cessna Citation 500 aircraft. The elevator angle is the manipulated variable in

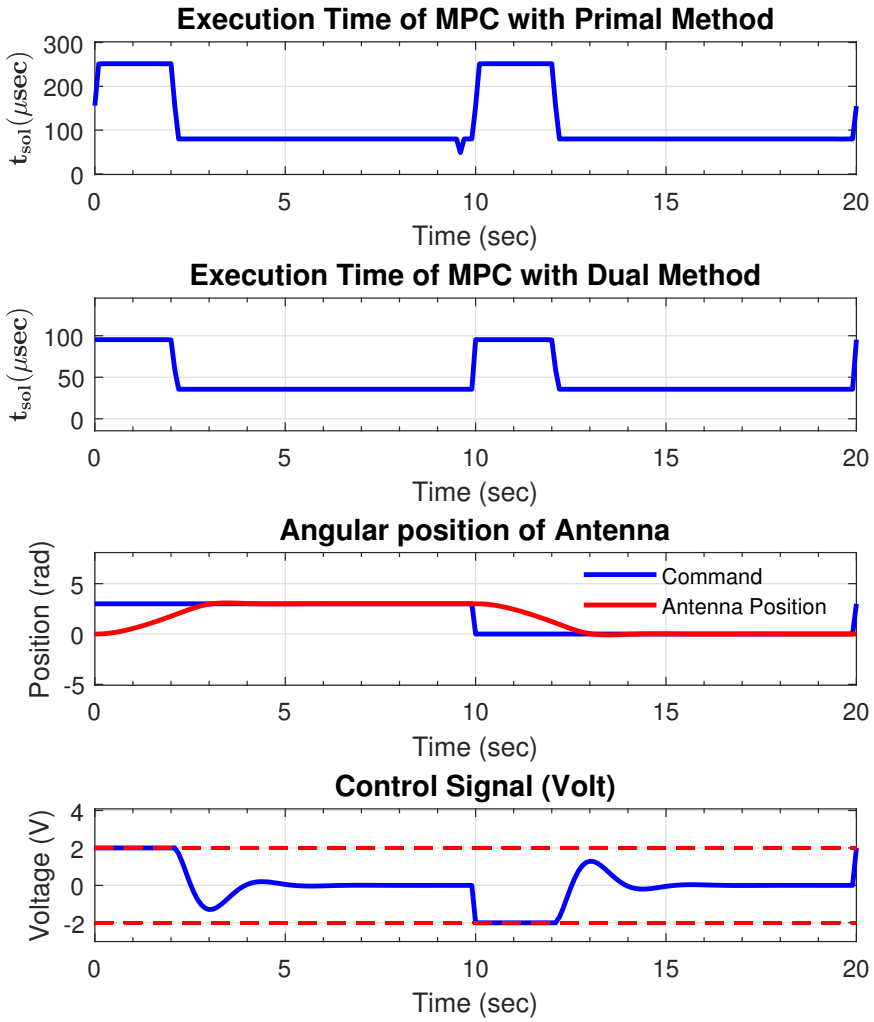

Figure 3.4: Performance result of the rotating antenna example based on the Dual Active Set Method. Figure also include the maximum execution time of the problem when it is solved by primal method.

the system ($u(rad)$), pitch angle ($y_1(rad)$) and altitude of the aircraft ($y_2(m)$), and altitude rate ($y_3(m/sec)$) are the outputs of the system respectively. The linearized continuous time state-space model is defined as

$$\dot{x}_{[4x1]} = \begin{bmatrix} -1.2822 & 0 & 0.98 & 0 \\ 0 & 0 & 1 & 0 \\ -5.4293 & 0 & -1.8366 & 0 \\ -128.2 & 128.2 & 0 & 0 \end{bmatrix} x_{[4x1]} + \begin{bmatrix} -0.3 \\ 0 \\ -17 \\ 0 \end{bmatrix} u \tag{3.29}$$

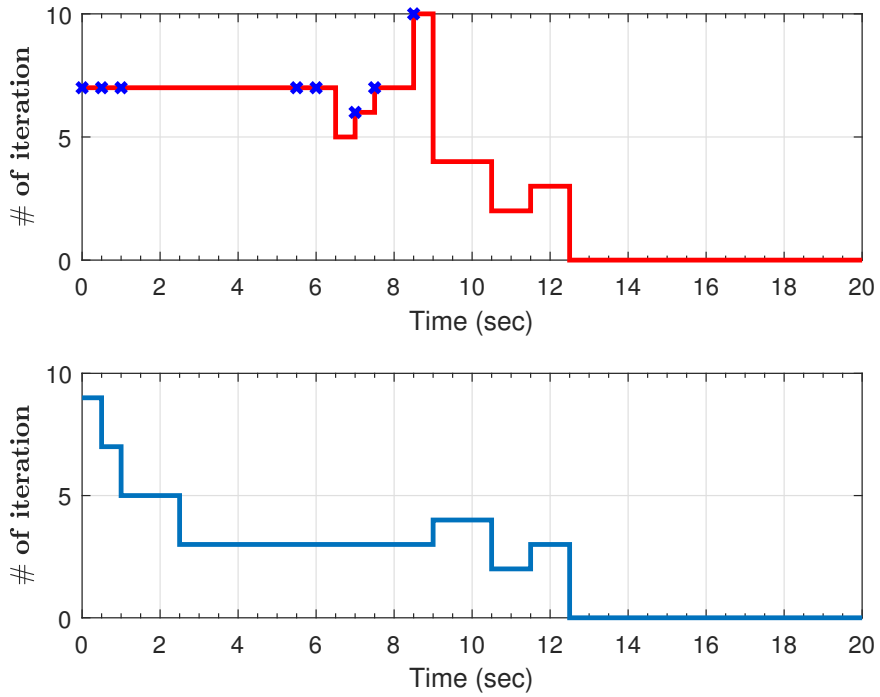

Figure 3.5: The number of iteration of Cessna example with scaled(bottom) and not scaled(top) results. The infeasible solutions are indicated by blue dots.

$$y_{[3x1]} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -128.2 & 128.2 & 0 & 0 \end{bmatrix} x_{[4x1]} \tag{3.30}$$

The CT model is discretized by taking the sample time as $T_s = 0.5$, the controller parameters for prediction horizons are $H_p = 10$, $H_u = 3$, and weightings are $Q = 1$, $R = 1$ in order to compare with the literature under the same operational conditions. The elevator angle and its rate, pitch angle and, altitude rate are constrained to:

$$|\Delta u| \leq 0.524 rad/sec \ (30^\circ/sec) \tag{3.31}$$

$$|u| \leq 0.262 rad \ (15^\circ) \tag{3.32}$$

$$|y_1| \leq 0.349 rad \ (20^\circ) \tag{3.33}$$

$$|y_3| \leq 30 m/sec \tag{3.34}$$

The Cessna Citation 500 example suffers from numerical errors in the single precision calculations. The Hessian matrix in the optimization formulation H is calculated based on the future predictions of the system that results to have huge power of the

state transition matrix, if the prediction horizon is large. In this example, H matrix is in the order of $1e7$ that causes numerical errors in the calculations. To overcome this issue, the original optimization formulation in 2.26 is re-formulated as,

$$\min_{z} \ \frac{1}{2} z^T \tilde{H} z + z^T \tilde{g}_k \tag{3.35a}$$

subject to

$$\tilde{W} z \leq \tilde{b}_k \tag{3.35b}$$

where

$$\tilde{H} = \alpha H, \ \tilde{g} = \alpha g \text{ and, } \tilde{W} = \beta W, \ \tilde{b} = \beta b.$$

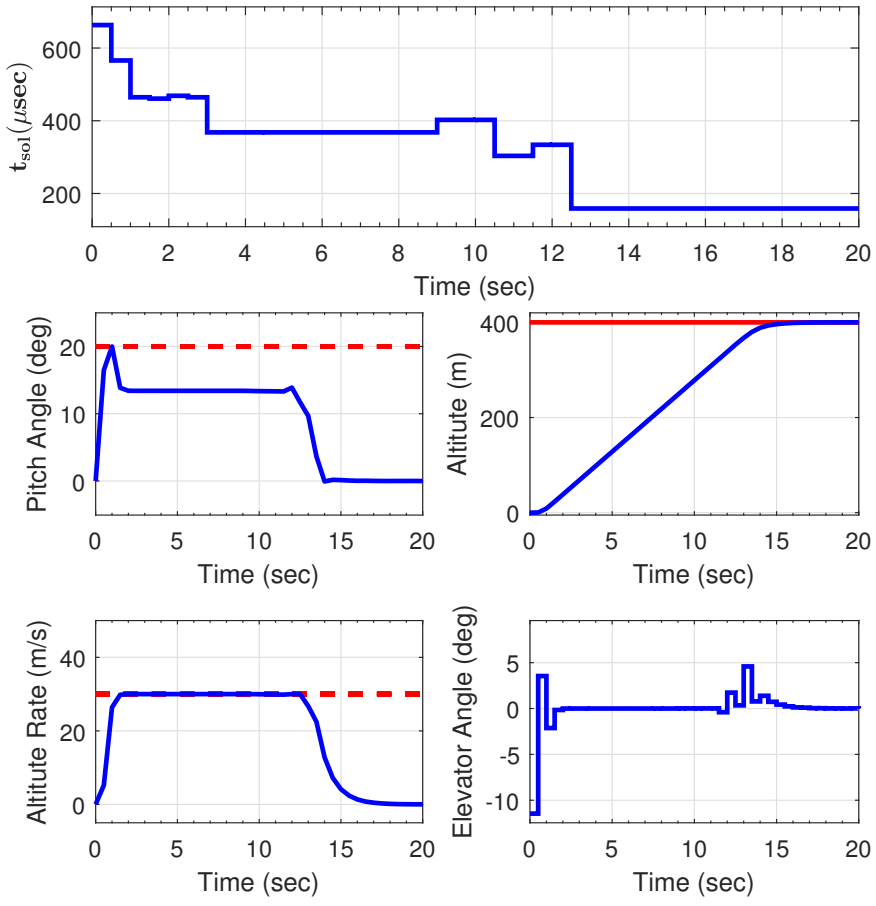

Figure 3.6: The execution time and overall results of the Cessna example. The maximum execution time($652\mu sec$) calculated at the maximum number of iteration occurred, which is the both pitch angle and altitude rate hit their limits.

The scaling of the original problem by using parameters α and β does not affect the solution but helps to increase the accuracy of the solution. The scaling parameters are usually chosen as to bound original problem in some range, e.g., ± 1 or ± 10. The results in 3.5 show that most of the optimization steps resulted in the infeasible solution. Scaling the original problem improves the accuracy of the solution where the numerical problems occur.

Fig.3.6 illustrates the overall results for execution time and responses of the aircraft. The Cessna aircraft follows the predefined setpoint($400m$) and satisfies the system's constraints. The maximum execution time is measured as $652\mu sec$, which is below among the studies in the literature, e.g., the best execution time is $1.10ms$ [31, 33, 34].

CHAPTER 4

IMPLEMENTATION OF MPC FOR ELECTRIC MOTOR: PMSM

Permanent magnet synchronous motor (PMSM) is one of the most common electric machines used in various industries due to its torque-speed characteristic and reliability for long-term use. One of the most common techniques to control the PMSM is vector control, also known as field-oriented control (FOC). Increased demand for high-performance operations in electrical drive systems has led engineers to adopt MPC and similar advanced model-based optimal algorithms for the control PMSM machines.

Several applications with model predictive torque control(MP-TC) successfully applied to control PMSM by computing the control signal via online optimization PMSM [15, 36]. In addition to these, some researchers also integrated field weakening operation with predictive control for some applications that require over-speed capabilities of the motors [37, 38, 39, 40, 41]. Mynar et al.[38] implement explicit model predictive control in high performance dual-core i7 processor that results in consuming a large amount of memory space. As a further investigation, the authors mention performing such an algorithm in the low-cost motion controller. Most of the other studies [37, 40, 41] adapted Finite Control Set (FCS) which basically finds the optimal switching position instead of producing the duty cycle as in Continuous Control Set(CCS) [38]. This approach couples the sample and switching frequency that causes to operate the inverter at variable frequency. In [42], unconstrained MPC is used to generate the d axis current reference and to satisfy the constraints, saturation blocks are utilized.

The scarce resources in online CCS-MPC with the field weakening operation and its implementation in low-cost platform encourage us to investigate this application. As

a further contribution, we address hints and some suggestions for the ease of practical implementation.

4.1 System Modeling

MPC is a state-space domain model-based control policy; thus, we need to derive a state-space representation for the PMSM system. We utilize the synchronous reference frame model (d-q model) because it makes the electrical variables stationary in the steady-state [19] and enables the regulation of torque and flux content separately (vector control) to achieve the maximum efficiency and perform the field-weakening operation. Following equations models the dynamics for a PMSM based on $d-q$ reference frame model [43].

4.1.1 Mathematical Model

For a PMSM, d-q reference frame model is given by the following equations [43] and q-axis and d-axis equivalent circuits are shown in Fig. 4.1 [2].

$$\frac{di_d(t)}{dt} = \frac{1}{L_d}(V_d(t) - R_s i_d(t) + \omega_e(t) L_q i_q(t)) \tag{4.1a}$$

$$\frac{di_q(t)}{dt} = \frac{1}{L_q}(V_q(t) - R_s i_q(t) - \omega_e(t)(L_d i_d(t) + \lambda_m)) \tag{4.1b}$$

$$T_e(t) = \frac{3}{2} pp(\lambda_m i_q(t) + (L_d - L_q) i_d(t) i_q(t)) \tag{4.1c}$$

where

λ_m	magnetic flux of the PM	Wb
$i_d - i_q$	*d axis*, *q axis* current	A
$V_d - V_q$	*d axis*, *q axis* voltage	V
R_s	stator phase resistance	Ω
$L_d - L_q$	*d axis*, *q axis* inductance	H
ω_e	rotor electrical speed	rad/s
T_e	electromechanical torque output	Nm
pp	number of pole-pairs	$-$

Finding a linear state-space approximation that can accurately capture the PMSM

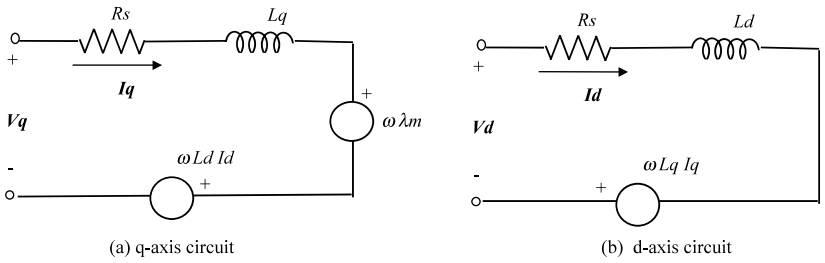

Figure 4.1: Permanent Magnet Synchronous Motor d-q axes Equivalent Circuits [2]

dynamics is critical for implementing (linear) MPC algorithms. In this study, we follow the approximation by Bolognani et al. [19] and treat the bilinear terms, $\omega_e(t)i_d(t)$ and $\omega_e(t)i_q(t)$, as new state variables and further assume that they are constant over prediction horizon. We also integrate the measured rotor speed as an exogenous input and again assume that it is constant over the prediction horizon. As a result, the state- and input-vectors of the linear state-space PMSM model takes the form $[i_d \; i_q \; \widehat{\omega_e i_d} \; \widehat{\omega_e i_q}]^T$ and $[V_d \; V_q]^T$ respectively. We finally discretize the CT state-space model via the forward-Euler method with sample time T_s to obtain the discrete-time state-space model structure as

$$x_{k+1} = Ax_k + B_u u_k + Gw_k \quad , \quad y_k = Cx_k \tag{4.2a}$$

$$A = \left[\begin{array}{cc|cc} 1-\frac{T_s R_s}{L_d} & 0 & 0 & \frac{T_s L_q}{L_d} \\ 0 & 1-\frac{T_s R_s}{L_d} & -\frac{T_s L_q}{L_d} & 0 \\ \hline \multicolumn{2}{c|}{\mathbf{0}_{2\times 2}} & \multicolumn{2}{c}{\mathbf{I}_{2\times 2}} \end{array}\right] \tag{4.2b}$$

$$B = \left[\begin{array}{cc} \frac{T_s}{L_d} & 0 \\ 0 & \frac{T_s}{L_q} \\ \hline \multicolumn{2}{c}{\mathbf{0}_{2\times 2}} \end{array}\right], G = \left[\begin{array}{c} 0 \\ -T_s\frac{\lambda_m}{L_q} \\ \hline \mathbf{0}_{2\times 1} \end{array}\right], C = \left[\begin{array}{c} \mathbf{I}_{2\times 2} \\ \hline \mathbf{0}_{2\times 2} \end{array}\right]^T$$

4.1.2 Field Weakening Operation

The torque generated by the PMSM given by (4.1c) has two sources: permanent magnet flux and rotor saliency. In a surface-mount PMSM, d-q axis inductances are almost equal to each other ($L_d \approx L_q = L$), and the torque is generated dominantly by the magnet flux and q axis current. Therefore, in the normal operation of PMSM, i.e., below-rated speed, d axis current is set to 0 ($i_d = 0$) to produce maximum torque per

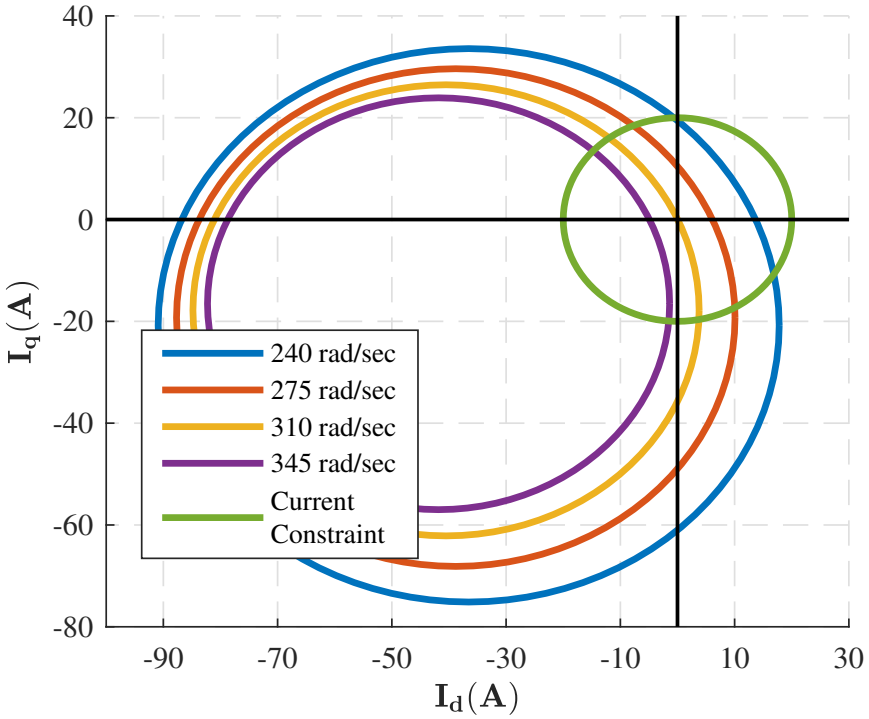

Figure 4.2: Current and voltage circles for different speed values of the PMSM.

amperes (MTPA). In order to exceed the rated speed, negative i_d values are used to limit the back EMF $(\omega_e L_d i_d + \omega_e \lambda_m)$ by creating an opposite magnetic flux to the permanent magnet in the field-weakening region. A given i_q^* reference can only be achieved if a certain amount of negative i_d^* current is applied. Applying negative i_d current to reduce the back EMF so that the PMSM can reach the speed beyond its rated speed is called a field-weakening operation. It technically modifies the current references (i_d^*, i_q^*) at the operating speed to get maximum torque output. A common way of vector control of the PMSM is realized by a voltage source inverter, which includes six transistors fed by a DC voltage source. Therefore, electrical sources are limited to the DC value of the voltage source (V_{max}) and the maximum current capability of the transistors (I_{max}). These limits are related to $d-q$ axis variables as

$$V_{max}^2 \geq V_d^2 + V_q^2 \quad , \quad I_{max}^2 \geq I_d^2 + I_q^2 \tag{4.3}$$

In the steady-state, derivative terms are equal to zero in (4.1a) and (4.1b). Rewriting (4.3) by substituting steady-state forms of (4.1a) and (4.1b) yields

$$\frac{V_{max}^2}{R_s^2 + L^2\omega_e^2} \geq (i_d + \frac{L\omega_e^2\lambda_m}{R_s^2 + L^2\omega_e^2})^2 + (i_q + \frac{R_s\omega_e\lambda_m}{R_s^2 + L^2\omega_e^2})^2 \tag{4.4}$$

The current equation in (4.3) represents the inner region of a circle whose center is at the origin and radius is equal to (I_{max}). This circle is called the *current circle*. On the other hand, the voltage equation represents the inner region of the *voltage circle*, for which the center and radius are equal to

$$\text{center} = \left(-\frac{L\omega_e^2 \lambda_m}{R_s^2 + L^2\omega_e^2}, -\frac{R_s \omega_e \lambda_m}{R_s^2 + L^2\omega_e^2} \right), \tag{4.5}$$

$$\text{radius} = \frac{V_{max}^2}{\sqrt{R_s^2 + L^2\omega_e^2}}, \tag{4.6}$$

respectively. The VSI cannot operate outside the current and voltage circles physically. Therefore, the operating point (i_d, i_q) should be inside the circles at any time. Note that the current circle is stationary while voltage circle parameters are functions of the rotor's electrical speed. To set a proper reference operating point (i_d^*, i_q^*) voltage circle equation should be dynamically calculated and the limit values should be updated. Fig. 4.2 illustrates the voltage and current circles for a PMSM with $I_{max} = 20A$ for different speed values. As shown in the figures, voltage circle shrinks as the speed increases and separates from the current circle after a critical speed value. $240\ rad/s$ rotor speed value approximately separates constant torque and constant power region. Voltage limit circle crosses the origin approximately at $310\ rad/s$ rotor speed, which is the theoretical no-load speed. It is inevitable to have $i_d < 0$ for higher speed demand than $310\ rad/s$, even if no load applies to the rotor shaft. The operating point for the PMSM should be inside the intersection of the two circles. Therefore, as the speed increases, maximum achievable i_q is limited below I_{max} value.

4.1.3 Model Verification

Even though the PMSM model is well-known and the producer provides its parameters, we developed a test bench to verify the motor model and its parameters for field weakening operation and model predictive control. The verification procedures are discussed in [44, 45]. The first experiment is to identify the value of the resistor and inductance for the $d-q$ axis. The rotor shaft is locked to prevent the rotation of the motor, i.e., avoid the back-emf so that we can find these parameters properly. Once we ensure that the motor is fully locked, we applied step-like voltage requests

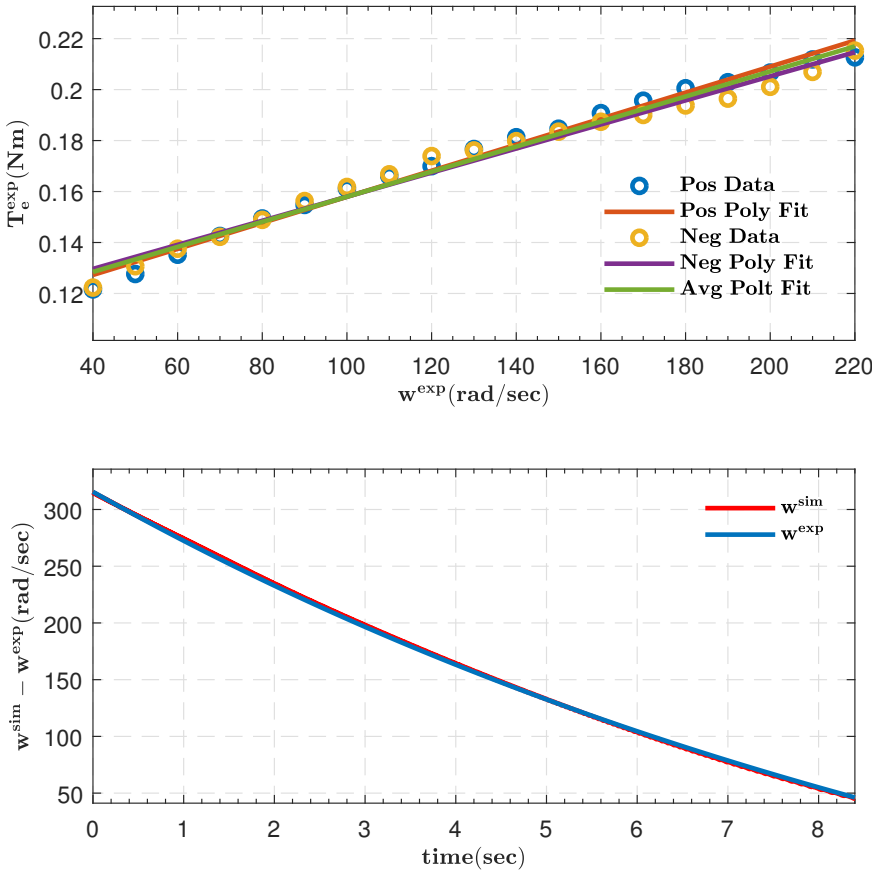

Figure 4.3: Top: The friction related with velocity and its polynomial approximation. Bottom: The operation part of the coasting down of the motor velocity to its stationary value.

for each axis separately. The resistor values for each voltage request are calculated in steady-state via DC component relation, i.e., $R = V/I$. On the other hand, the inductance values are found via the following equation,

$$i(t) = \frac{V}{R}(1 - e^{-\frac{t}{\tau}}), \text{ where } \tau = \frac{L}{R} \tag{4.7}$$

Time constant values are extracted from the data, i.e., time change from zero to 63% of the steady-state value to find the inductance values. Since the main aim is to create the motor model for performing the simulations so that control algorithms run perfectly before experimental testing, the mechanical parts of the model are also considered. Therefore, their parameters are identified via different methods. We start with identifying the friction of the overall system that consists of the friction in the bearing of the motor(mainly at a negligible level) and the connection with the rotor

shaft with dynamo. Since our application is to perform the field weakening, we only take care of the dynamic relationship between the velocity of the rotor and the friction torque. The following equation represents the general formulation of the mechanical side of the PMSM.

$$\frac{dw(t)}{dt} = \frac{1}{J}(T_e - bw - H_0 - T_{load}) \tag{4.8}$$

where J represents inertia, T_e and T_{load} torques are electrical input and load torque respectively. The other terms b and H_0 are the friction terms to represent viscous and Coulomb frictions. In order to eliminate the effect of inertia, we collect the torque values at steady-state velocity. Starting from the $40(rad/sec)$ to $220(rad/sec)$ velocity, the torque values are saved to derive the relation with velocity. We utilize first-order polynomial fit to represent the overall friction with the combination of static and dynamical relation. We derived the relationship for both positive and negative sides, as in Fig.4.3, and taking the average of them to be applied in the model. Once we have the friction terms, inertia is the only left term to be found. In this case, we apply the torque to the system to reach its maximum velocity. Then, we unplug the motor from the power source to coast down the stationary value by itself, i.e., eliminating the T_e term from the equation. The time-domain solution of the first-order equation of the overall mechanical side equation by extracting T_e is provided via the following equation,

$$w(t) = (w_0 + \frac{H_0}{b}e^{-\frac{b}{H}t}) - \frac{H_0}{b} \tag{4.9}$$

The Fig.4.3 illustrates the coasting down of the motor for both simulation and experiment after finding the H term via MATLAB/Optimization Toolbox. Once we ensure that the parameters of the motor are obtained separately, we apply varying q axis and constant d axis voltages to ensure that the overall model matches the actual motor. The performances of the open loop responses of the model for both simulation and experiment are provided via Fig.4.4

4.2 Controller Design

The essential feature of the MPC over standard PI-FOC handles constraints explicitly without using any add-on structure. It combines the two different controllers into a

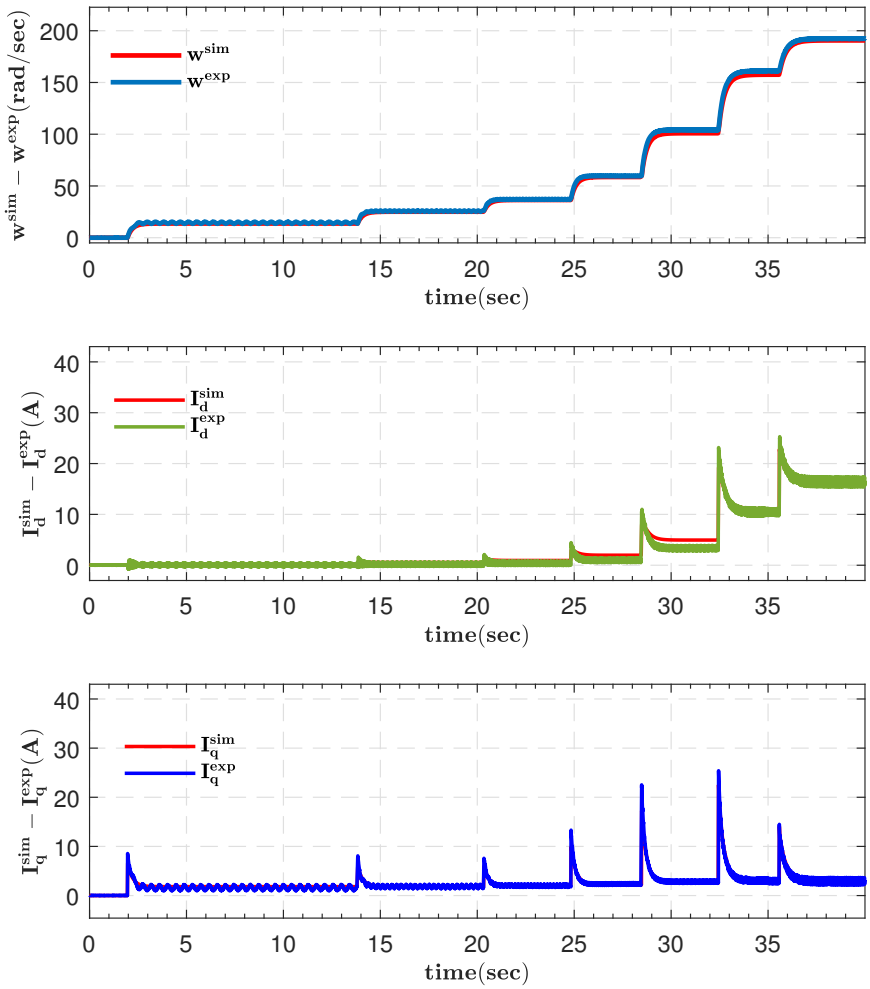

Figure 4.4: Overall performances of the open loop responses of the model for both simulation and experiment.

single loop and produces optimal control inputs to the system. The standard structure of FOC utilizes a cascaded loop, which includes the current loop in the inner while speed is in the outer loop. The standard architecture is preserved by placing MPC in place of the PI controller with its components in the current loop since the main bounds in PMSM come from voltage and current limitations. The MPC follows the references provided by the speed loop for the q axis and sets zero for the d axis to imply the MTPA when the motor operates below its rated speed.

The algorithm is also responsible for dynamically generating the required d axis reference by monitoring q axis current and motor velocity to perform field weakening operation, which enables to produce MTPA at beyond rated speed. Fig. 4.5 illustrates

the overall proposed structure that we implement for the CCS-MPC field weakening operation.

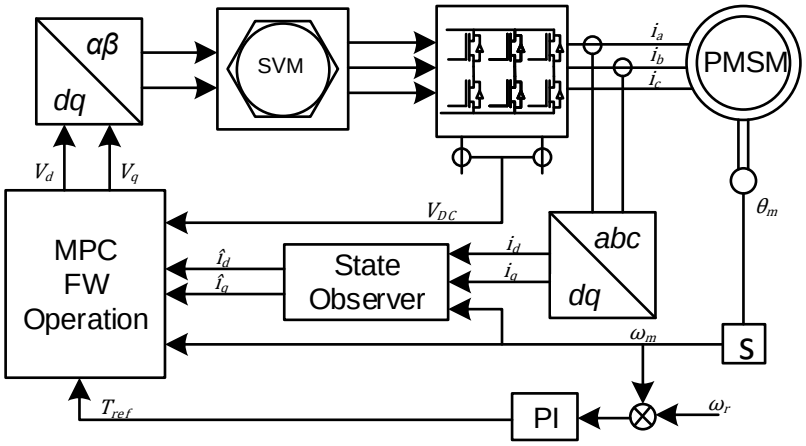

Figure 4.5: The schematic of the proposed structure for field weakening operation.

4.2.1 Constraints

Unlike the classical linear controllers, MPC does not require add-on implicit structures to handle the system's constraints. Under the MPC umbrella, the controllers explicitly address linear convex constraints (equalities and inequalities) on state variables, system outputs, and control/input signals in the system representation. In PMSM drive applications, upper and lower limits on *voltage* and *current* variables are the most dominant type of adopted constraints. The supply voltage on DC-link enforces a maximum value on the voltage supply to the drive, and it is $V_{max} = V_{DC}/\sqrt{3}$ for space vector modulation [46]. The peak stator current determines an upper bound constraint on the stator current variable.

We can transform the *voltage* and *current* constraints to 2-norm condition in (d, q) axis plane, such that;

$$\begin{aligned} x \in \mathbb{R}^2 \text{ s.t. } ||x||_2 < I_{max} \\ u \in \mathbb{R}^2 \text{ s.t. } ||u||_2 < V_{max} \end{aligned} \tag{4.10}$$

We apply two polygonal approximations to transform these two quadratic constraints into linear form to adapt them into the MPC framework as well as reduce the imple-

mentation complexity [19].

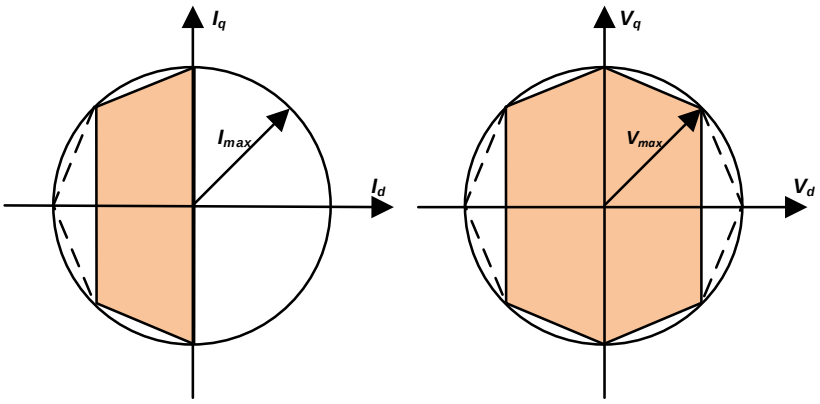

Figure 4.6: The schematic of voltage and current constraint which depends on the linear approximations

As for the voltage limit, we adopt an octagonal shape that combines the two constraint lines into one in the d axis direction. Since the q axis is responsible for generating torque to meet the load torque in the motor shaft, it is undesirable to give all DC-link voltage only for the d axis.This approximation has an acceptable level of underestimation, 70% of the maximum voltage value for the d axis. We can formally define the voltage constraint using the following equation;

$$\begin{bmatrix} -\frac{1}{\sqrt{2}+1} & 0 & -\frac{1}{\sqrt{2}+1} & \frac{1}{\sqrt{2}+1} & 0 & \frac{1}{\sqrt{2}+1} \\ 1 & -\frac{\sqrt{2}+2}{\sqrt{2}+1} & -1 & -1 & \frac{\sqrt{2}+2}{\sqrt{2}+1} & 1 \end{bmatrix}^T \begin{bmatrix} V_d \\ V_q \end{bmatrix} = \begin{bmatrix} V_{max} \end{bmatrix}_{6x1} \tag{4.11}$$

We modify the constraint adaptation above for the current implementation by cropping out the polygon's right half-plane since the only negative current of the d axis is used for field weakening operation. Fig. 4.6 illustrates the schema for these two constraints. The constraint on current according to the figure is given by the following equation;

$$\begin{bmatrix} -\frac{1}{\sqrt{2}+1} & 0 & -\frac{1}{\sqrt{2}+1} \\ 1 & -\frac{\sqrt{2}+2}{\sqrt{2}+1} & -1 \end{bmatrix}^T \begin{bmatrix} I_d \\ I_q \end{bmatrix} = \begin{bmatrix} I_{max} \end{bmatrix}_{3x1} \tag{4.12}$$

In practical application, it is highly suggested setting the voltage limit to DC-link voltage measurement, if available, in every step for the case of any change in voltage that may be different from its constant value.

4.3 Results

The MPC with developed QP Solver algorithm has been verified in normal, Processor in the Loop(PIL) simulations and experimentally tested on the custom-made PMSM. The proposed controller structure is implemented in both $C2000$ for PIL simulation and $C6000$ for the experimental testing.

Table 4.1: PMSM and CONTROLLER Parameters

Parameter	Value
(J, B)	$(6.10^{-3}kgm^2, 49.10^{-5}Nm/(rad/s))$
λ_m	0.0106 Wb
R_s	120 $m\Omega$
$L_d \approx L_q$	220 μH
Torque Constant	0.09 Nm/A_{rms}
pp	4
T_s	200μs
(H_p, H_u)	$(4, 2)$
(Q, R)	$(\mathbf{I}_{2\times2}, \frac{1}{20}\mathbf{I}_{2\times2})$
(P, I)	$(2, 0.5)$
V_{max}	$24/\sqrt{3}$ V
I_{max}	20 A

Both processors have floating-point capability provided by Texas Instruments. Having cost-effective products and being widely used in many industries, especially for motion control, are the main reasons to choose the Texas Instruments processor family for this work. The MPC with QP solver algorithm is efficiently implemented in plain C code and has been carried out in $F28377S$ and $C6713B$ to ensure the consistency of our implementation in different types of processors with different specifications. The MPC design parameters, along with all the related motor parameters, are listed in Table 4.1. Since the algorithm uses a cascaded structure, two different sample times are scheduled; $T_s^{fast} = 200\mu s$ is for the fast loop, i.e., current controller, and $T_s^{slow} = 1ms$ is for the slow loop, i.e., speed controller.

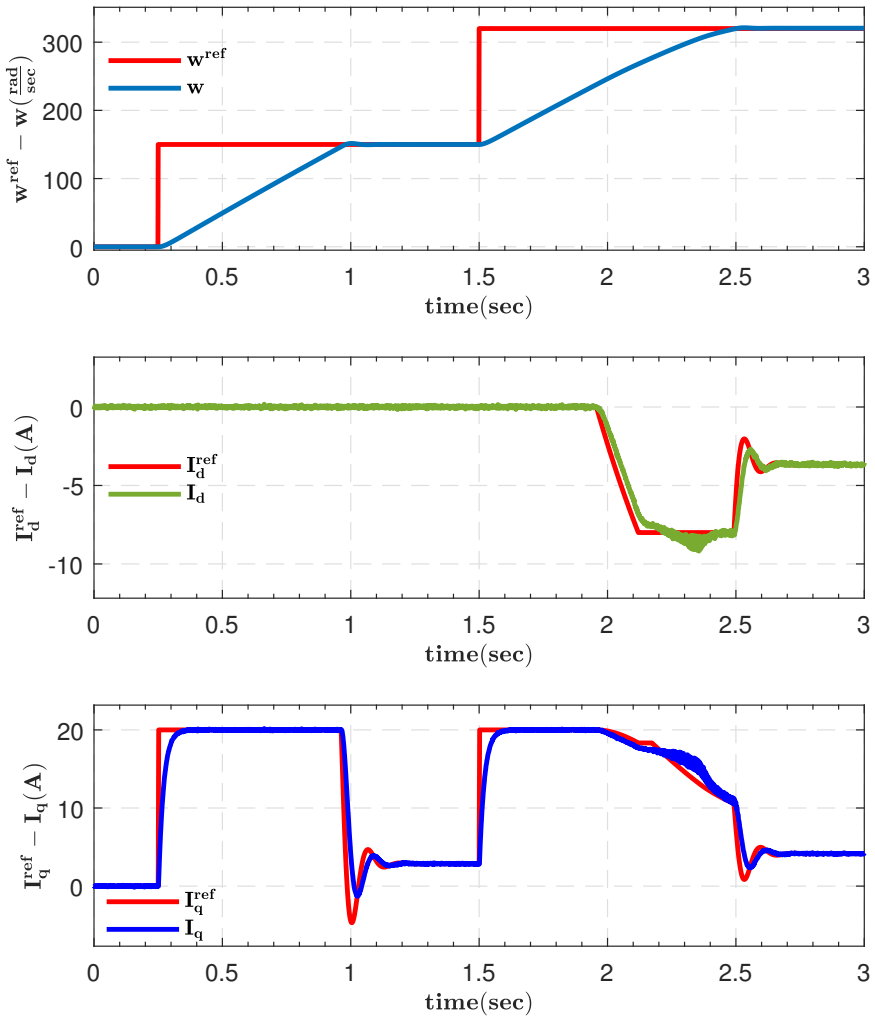

Figure 4.7: PIL simulation results of tracking performances for speed and current references. The i_d current enters the scene after $t = 2s$ to weaken the flux that enables the motor tracks the desired speed value. Between $t = 2 - 2.3s$ i_q does not perfectly track the desired value because of linear approximation on the constraint. The algorithm regulates the voltage value to satisfy the constraints.

4.3.1 PIL Simulation Results

Prior to performing experimental testing, the PIL simulation of the proposed algorithm is carried out to analyze the efficiency of the implementation in terms of execution speed and memory usage. PIL simulation environment enables debugging the code easily and provides the execution speed of selected blocks by using a related CPU timer.

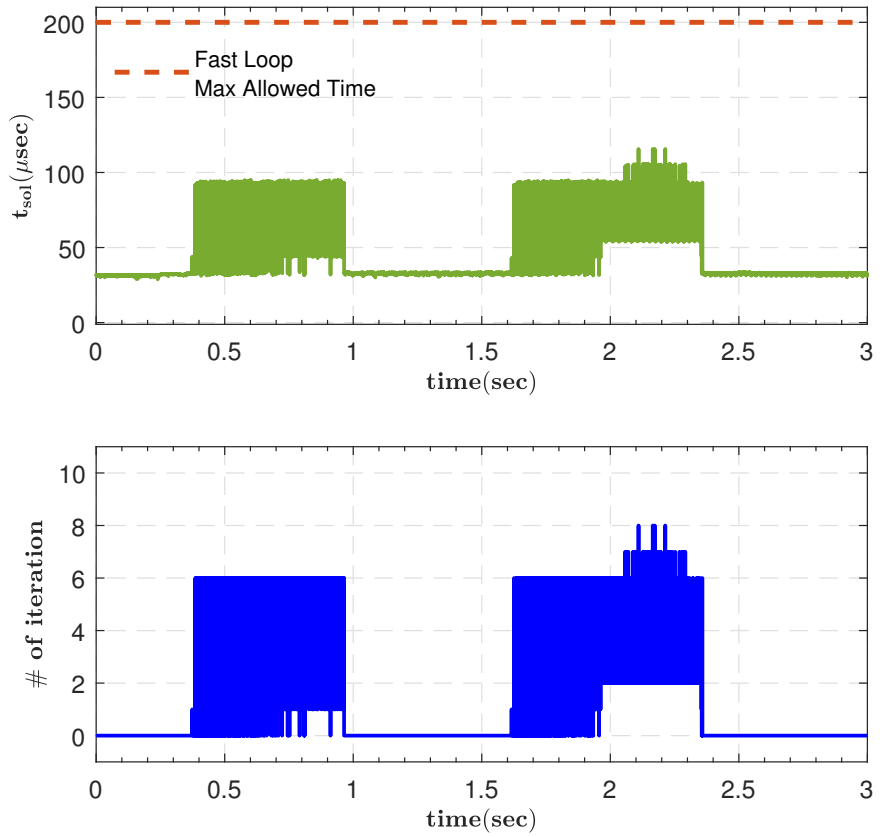

Figure 4.8: Execution time is strictly less than the maximum allowed time through the operation in PIL simulation. The execution time and the number of iterations are consistent except at a point at which the maximum overshoot on i_d current occurs.

Thus, it is possible to detect the code part where the main computational burden of the algorithm lies. MATLAB/Simulink environment is chosen to perform PIL simulation in the $F28377S$ processor. The main controller algorithm runs in the processor, as the simulation part involves the PMSM model and the desired set point for the speed loop. Simulation and the real-time environments communicate through serial connection. Since the main reason for PIL simulation is to provide that our implementation is feasible, only the tracking of speed and current loop and the success of fulfilling constraints are evaluated. Fig. 4.7 illustrates the tracking performances of the proposed algorithm for both speed and current loop. The maximum reachable speed of the PMSM with the supplied voltage level is $310\ rad/s$, the i_d current enters the scene at $t = 2s$ to weak the flux to achieve the desired speed set point, i.e., $320\ rad/s$.

Fig. 4.8 shows the timing and the number of iterations that the controller executes.

The execution time always lies under the predetermined sample time. As an additional note, the execution time in PIL simulation does not include the ADC readings, their parsing process, and the DAC step, which are the necessary parts for real-time application. In addition, the algorithm consumes only $3.5kB$ out of $164kB$ memory space that demonstrates the feasibility in terms of using minimal memory.

4.3.2 Experimental Results

After successful demonstration of the proposed algorithm via PIL simulations, experimental testing is carried out. The processor which executes the algorithm is now moved to the $C6000$ series, i.e., $C6713B$, to perform the algorithm. Fig. 4.9 shows the test bench that is used throughout the experimental testing step. It includes

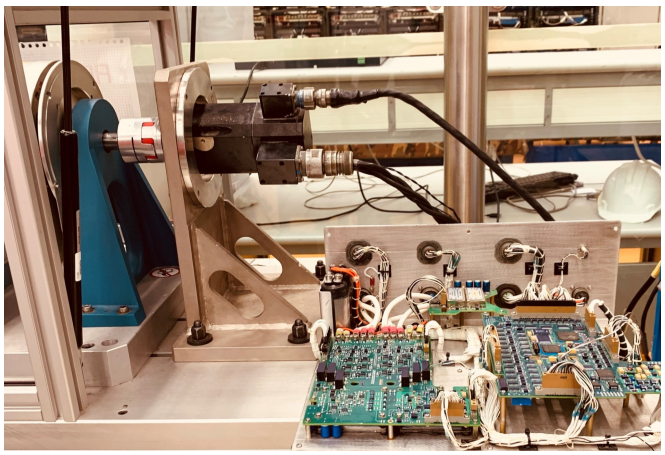

Figure 4.9: Test bench used in experiment of field weakening operation. It consists of PMSM connected to the dynamo and the custom-made driver unit.

the dynamo, which is connected to the motor shaft and the motor driver unit. We measure the motor's position via a resolver attached to the rotor shaft. The velocity is obtained by using a resolver to digital converter(RDC) that performs the feedback structure to generate the velocity rather than directly taking the position's derivative. The driver unit is custom-made and has the ability to be replaced with a different processor according to the requirements. Fig. 4.10 presents the $d-q$ axis voltages and overall tracking performances of the algorithm for speed and current loops. Our experimental scenario split the reference angular velocity signal into two regions to

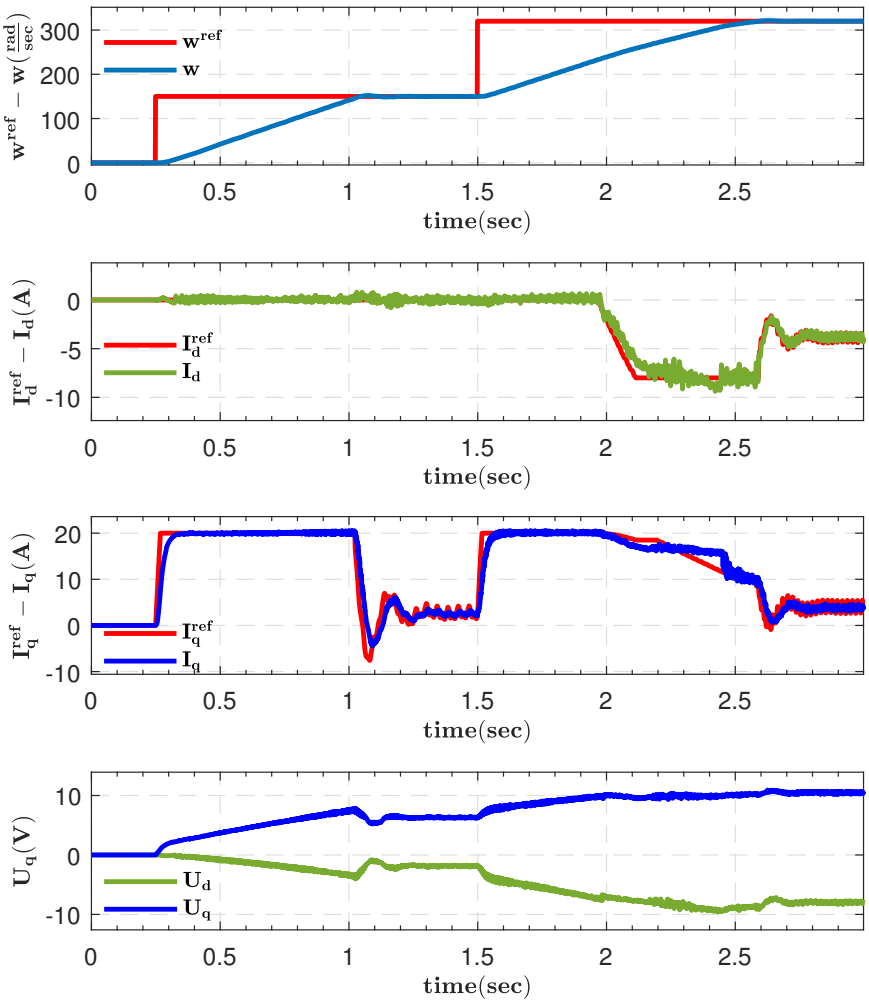

Figure 4.10: Tracking performances of speed and current loops and the voltages which are generated from online MPC in experimental testing.The tracking performances are successfully fulfilled and are similar with the simulation results thanks to well-constructed plant model

evaluate the "constant"-torque and "constant"-power regions. We feed step-input type reference signals in both zones to push the motor to operate around its limits (torque and power) and stressing the optimization solver. Specifically, at $t = 0$, the system starts at initially at rest condition, and at $t = 0.25$s, we supply the combined cascaded control algorithm a constant angular velocity reference signal of $\omega_{ref} = 150\text{rad/s}$ until $t = 1.5$s, where we jump the reference signal to $\omega_{ref} = 320\text{rad/s}$. In the first zone, i.e. $t \in (0.25, 1.5)$s, the motor dominantly operates at the torque limit (where q axis current is constant) until the motor velocity reaches to the desired value (at $t \approx 1.0$s), and thus angular velocity increases almost linearly during this period. In

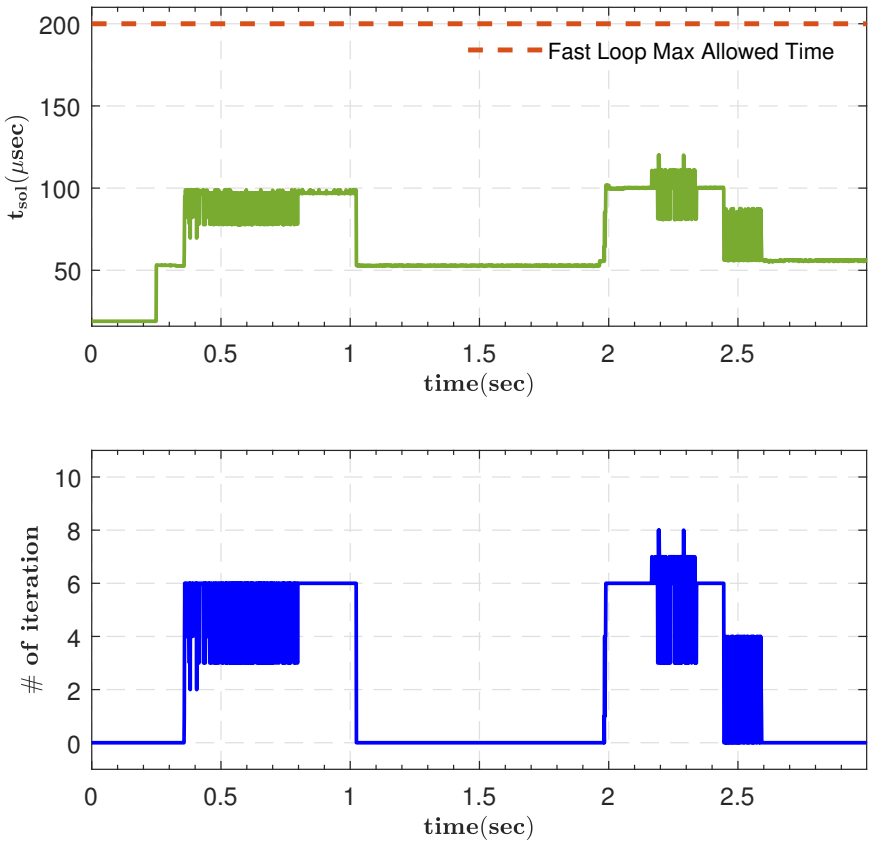

Figure 4.11: Execution time and the number of iteration in the real time experiment to generate the suitable voltages for both axis while satisfying constraints.

the second region, where the desired angular velocity is $\omega_{ref} = 320\text{rad}/s$, the behavior of the motor and controller is similar to the first zone until around $t \approx 2\text{s}$, where at that point field weakening operation activates. During this period, the algorithm introduces a negative d axis current that reduces q axis current to respect the maximum torque per ampere criteria. We observe the effect of the linear approximation on the current constraint, especially in the q axis, after $t \approx 2\text{s}$ since the algorithm puts extra effort to satisfy the constraints that reduce tracking performance. Once the motor angular velocity reaches its final desired value, the algorithm applies consistent $d-q$ axis currents to keep the motor at the desired speed and compensate for the friction. Fig. 4.11 illustrates the execution time in the experimental test that demonstrates feasibility since computation time always stays inside the sampling time during the operation. One should also note that the execution time also includes the ADC readings, their parsing process, and the DAC phases. In addition to execution time, our

implementation is also feasible in terms of using minimal memory. The memory occupancy of both algorithm and necessary QP data is only $6kB$ out of $192kB$ in $C6713B$.

We also evaluate the state dependencies between $d-q$ axis current and voltage by picturing the actual motor data to illustrate the performances based on our linear approximation of the constraints. Fig. 4.12 presents the linear approximation polygons on voltage and current constraints together with real-time experiment data. Apart from the minor deviations on the lines because of noisy measurements and process uncertainty, the electrical state's values always respect the linear constraints.

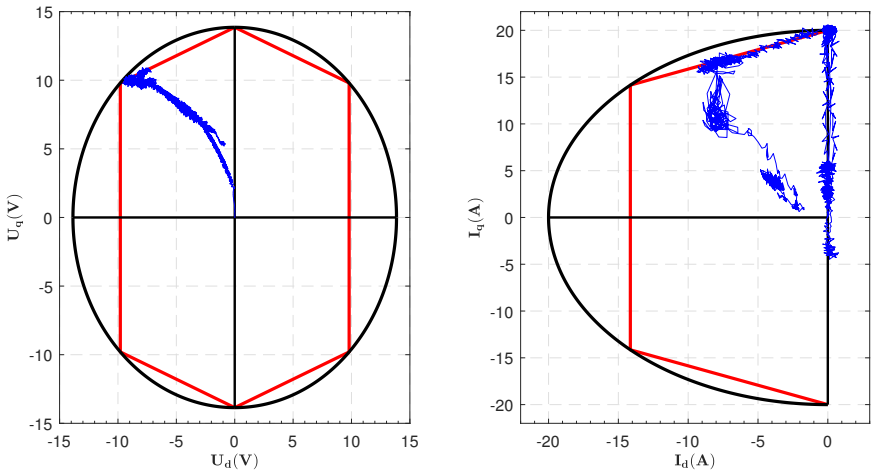

Figure 4.12: The real time experiment values of both voltages and currents lie inside the linear approximation polygons that represents the circle. There is small deviation on the edge of the linear approximations because of the noise level in our measurements.

CHAPTER 5

IMPLEMENTATION OF MPC FOR GIMBAL PLATFORM

Gimbal platforms consist of a set of independent orthogonal axes that can rotate or orient with respect to the base on which it is mounted. It has a wide range of usage from military to industrial applications such as surveillance, gun-turret platform, target tracking or air-defense purpose on the military side while pointing application on the telescope, mirror stabilization or drone's camera for smooth recording on the civilian usage side [47, 48]. Its usage usually requires more than one axis to track a target or point to the desired object. The demand to increase the performance of such platforms is more intensive since increased requirements on the precise pointing and long-distance target tracking application. An example of such a platform for tracking purposes is illustrated in Fig. 5.1.

Figure 5.1: The 2 axes gimbal platform to be used in target tracking application (Photo Courtesy of ASELSAN Inc.).

The most popular implementation of such platforms is to stabilize the platform against disturbances so that it keeps its direction without any deterioration on the pointing of the object. For this purpose, the measurement with respect to the constant inertial frame is required. The gyroscope provides the orientation or angular velocity with respect to the inertial frame so that the platform can hold itself constant by means of the gyroscope. In this application, we examine the platform in terms of increasing the set-point tracking performance by comparing it with one of the well-known classical controllers.

5.1 System Modeling

The crucial part in the development of MPC is to provide the system model that captures the overall system dynamics and possibly representing them in a simple manner. In this context, we assume that the traverse axis of the gimbal is a rigid body and ignoring the flexible structures that reveal at higher frequency in resonance and anti-resonance form. We represent the mathematical model of the traverse axis of the gimbal platform via a first-order differential equation and given with the following equations.

$$T_e = J\dot{w} \tag{5.1}$$

To ensure that the basic model is sufficient to represent the system dynamics, we also create a test bench to identify the system model via input-output data. The system identification also provides additional information from the system, such as delay or flexible body structures (i.e., determining whether it is ignorable). The key point in data-driven system identification is the richness of the input-output data. The input signal should be rich enough to excite different system characteristics, making them observable at the output signal. In this context, we apply a sine wave input signal to the system whose amplitude is constant, and frequency is swept from 1Hz to 120Hz. A critical point in the concatenation of the different frequency sinusoidal signals is to ensure the smoothness and continuity of the input. To achieve this, we adjusted the duration of each sine wave, ensuring that an integer number of periods is completed before switching to the subsequent frequency. This approach provides a way to ana-

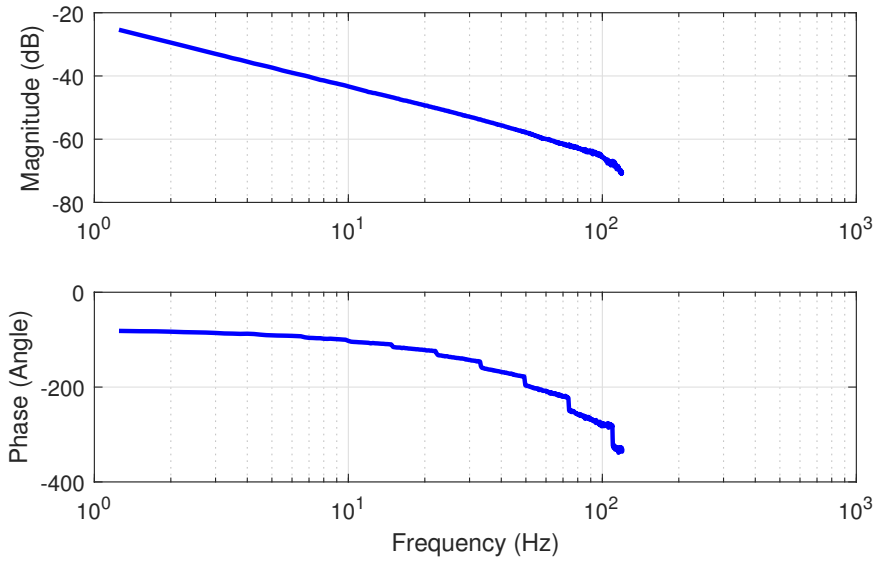

Figure 5.2: The Bode plot representation of traverse axis of gimbal platform calculated via input-output data.

lyze the system response in the frequency domain. The system response according to our input signal is presented in Fig. 5.2 as Bode plot.

According to the Bode plot, the characteristic of the system approximately matches with its mathematical model, i.e., representing the traverse axis as a rigid body. The inertia term is calculated from low frequency amplitude value of the Bode plot. On the other hand, the phase plot cannot be exactly represented via a rigid body approach since it decreases dramatically as the frequency increases. It is because of the delay term in the model that might result from group delay at the measurements and the delay at the driver unit. We approximate the overall system dynamics via equation 5.2 by taking the tradeoff between the accuracy and the computational complexity of the system into account.

$$T_e(s) = G_{rigid}(s)e^{-st} \tag{5.2}$$

The system response in Fig. 5.2 does not include the friction effect, which reveals especially at a lower speed and dominates the system response due to stick-slip phenomena. The MPC can handle the friction term by including it in the system model to eliminate its effect on the system. Several studies in the literature model friction and eliminate its effect via several approaches [49, 50, 51]. The most dominant part of

the friction term occurs at a point that breaks away from zero velocity, namely static friction. The friction is the function of the velocity, and the friction effect is almost linearly getting higher with the velocity of the system, known as viscous friction. The transition between the static and dynamic part of the friction is represented via the well-known Stribeck model in the literature. Equation 5.3 represents the mathematical model of the complete friction that covers static, viscous, and Stribeck effects.

$$T_f(w) = T_c + (T_s - T_c)e^{-w/w_s} + T_w w \tag{5.3}$$

where T_c, T_s and T_w are Coulumb, static and viscous frictions respectively and the w_s is the Stribeck velocity. We performed various experiments in the gimbal platform to determine the parameters in the friction equation. We drive the gimbal platform under constant speed starting from $0.2deg/s$ to $10deg/s$ by collecting large measurement samples to obtain a well-approximation of the friction model. We repeated the verification test at different speed values for negative and positive directions to increase the repeatability. We observed from the measurements that the friction torque values for both directions are slightly equal to each other. Thus, the overall model is only considered for only one side of the velocity.

The mathematical model of friction is highly nonlinear. We shall rearrange the nonlinearity via linear approximation to include in MPC. We can approximate the nonlinear curves of the friction by the combination of two linear by making the tradeoff between accuracy and speed [52]. Fig. 5.3 illustrates the approximation of the friction together with its original curves. The two lines intersect at w_b velocity in which maximum accuracy lost occurs. These lines can be obtained via first-order Taylor series expansion of the original equation. The two lines on positive sides are derived via the following equations.

$$\begin{aligned} y_1^+ &= T_{f^+}^1(w) = T_s + \frac{\partial T_f(w)}{\partial w}\Big|_{w=0} \\ &= T_s + (T_v - \frac{T_s - T_c}{w_s})w \end{aligned} \tag{5.4}$$

$$\begin{aligned} y_2^+ &= T_{f^+}^2(w) = T_f(w_{max}) + \frac{\partial T_f(w_{max})}{w}(w - w_{max}) \\ &= T_f(w_{max}) + (T_v - \frac{T_s - T_c}{w_s}e^{-w_{max}/w_s})(w - w_{max}) \end{aligned} \tag{5.5}$$

The resulting approximation leads to describe the friction term as partial piecewise

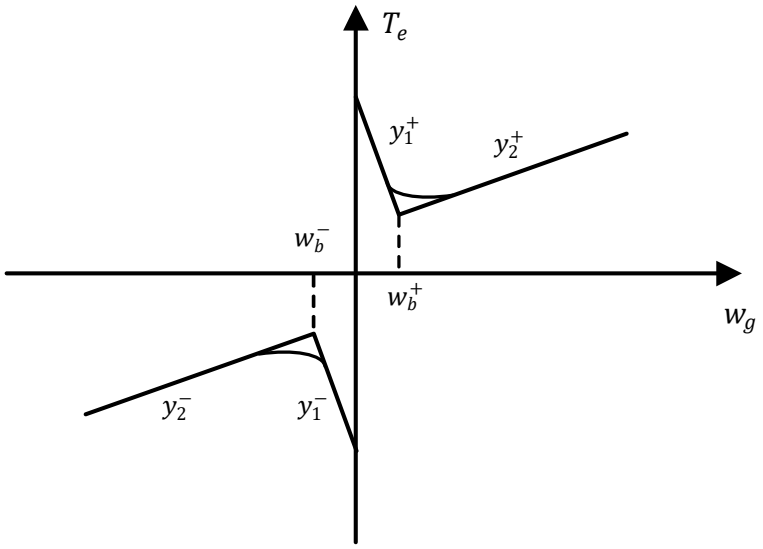

Figure 5.3: The Bode plot representation of traverse axis of gimbal platform calculated via input-output data.

function via following,

$$T_f = \begin{cases} a_1 + b_1 w & \text{if } |w| \in (0, w_b] \\ a_2 + b_2 w & \text{if } |w| \in (w_b, w_{max}] \end{cases}$$

The parameters of the linear lines are determined via Matlab/Optimization Toolbox. The result of the system identification based on the Bode plot in 5.2 together with the identified linear model is illustrated in Fig. 5.4.

Since the state-space model is required to construct the MPC, the discrete-time model that contains an only rigid body with sample time Ts is obtained via Forward Euler approximation, and it is given in the following equations,

$$\dot{w} = \frac{T_e - T_f}{J} \tag{5.6}$$

$$\dot{w} \approx \frac{w_{k+1} - w_k}{T_s}$$

$$w_{k+1} = w_k + \frac{T_s}{J}(T_e - T_f) \tag{5.7}$$

The delay term is the missing part in the discrete state-space model. There are several approaches to include the delay term in the linear state-space model:

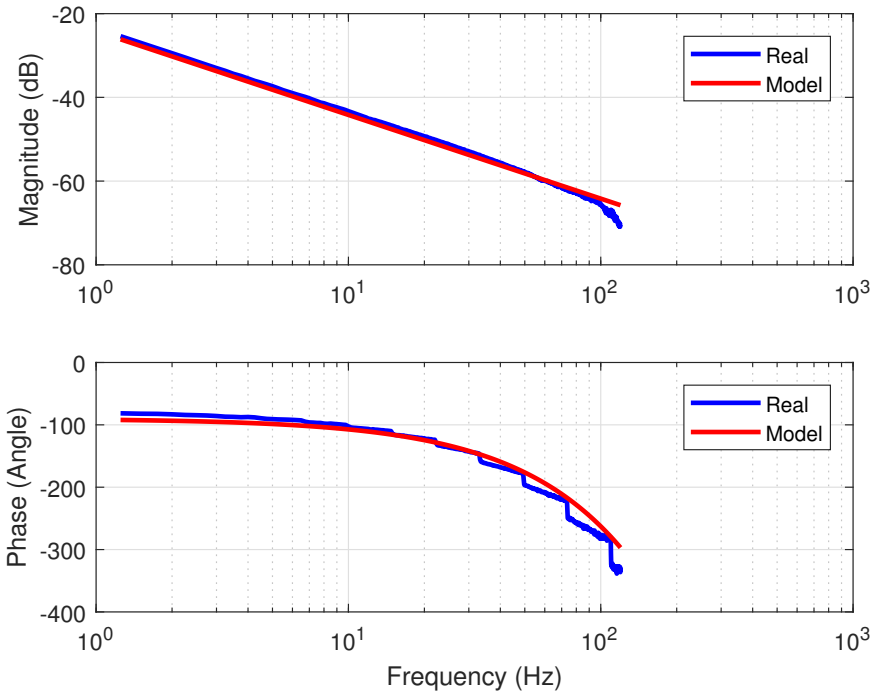

Figure 5.4: The Bode plot representation of model and real data taken from traverse axis of gimbal platform.

- The easiest way is to ignore the delay term in the design process if it is not dominant according to sample time. However, it shall be included in the analysis since it is essential to investigate the effect of the delay term in the stability analysis.

- The second method is to utilize the Pade approximation for the delay term. It is worth noticing that the prediction horizon should be long enough to include the delay dynamics in the Pade approximation since the resulting system has a non-minimum phase. This results in undershoot at the beginning of the response that confuses the controller if it does not have information on the future characteristic [53].

- The last method is to approximate the delay term as a delay buffer in discrete domain, as illustrated in Fig. 5.5. In this way, the system holds the memory of the past control signal during the delay interval [54].

The overall system model with delay dynamics in the discrete state-space form is

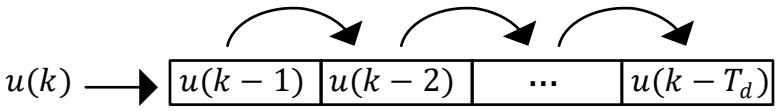

Figure 5.5: The delay buffer in discrete domain.

given in the following equations,

$$x_{k+1} = Ax_k + \begin{bmatrix} B & 0 & \dots & 0 \end{bmatrix}^T u_k^d \tag{5.8}$$

$$u_{k+1}^d = A^{del} u_k^d + B^{del} u_k \tag{5.9}$$

where

$$A^{del} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix}, B^{del} = \begin{bmatrix} 0 & \dots & 0 & 1 \end{bmatrix}^T \tag{5.10}$$

5.2 Controller Design

The essential feature of the MPC over classical controller handles constraints explicitly without using any add-on structure. It combines the two different controllers into a single loop and produces optimal control inputs to the system. The standard servo structure utilizes a cascaded loop, including the current loop in the inner while speed is in the outer loop. The standard architecture is preserved by putting MPC in place of the PI controller with its components in the speed loop. The dynamics of the current loop are also included in the MPC formulation since it has a series connecting to the system model. Note that the system identification in Fig. 5.2 determines the system model via measured torque and velocity from the system that excludes the motor dynamics. The dynamics of the current loop(closed-loop) are in the form of a first-order low pass filter with a cut-off frequency at $100Hz$(validated via simulation and verified via an experimental test). The final system model from the point of view

of the speed loop is formulated by cascading torque loop dynamics and 5.8,

$$x_{k+1}^f = A^f x_k^f + B^f u_k^f + G^f w \tag{5.11}$$

$$y_k^f = C^f x_k^f \tag{5.12}$$

where $x_k^f \in \mathbb{R}^7$ and G is the measured disturbance generated from the friction model. The controller excludes the friction model so that the MPC generates the control signal based on the standard form. The MPC includes the friction model at the end of the control signal to eliminate the friction effect. Note that we determine the maximum torque limit in each step based on the friction value that is generated from velocity.

The requirement from the MPC for the gimbal platform is to behave as a regulator while respecting the constraints. In this context, we utilize the MPC formulation for tracking requirements that also adopted for the PMSM example. Thus, the optimal control input for the gimbal platform in each step is obtained as the solution of the following formulations.

$$\min_{\Delta u} \sum_{i=1}^{H_p} ||Q^{\frac{1}{2}}(y_{k+i} - r_k)||_2^2 + \sum_{j=0}^{H_u-1} ||R^{\frac{1}{2}} \Delta u_{k+j}||_2^2 \tag{5.13a}$$

$$\text{subject to: } x_{k+i+1} = Ax_{k+i} + B_u u_{k+i} \tag{5.13b}$$

$$y_{k+i+1} = Cx_{k+i+1} \tag{5.13c}$$

$$u_i^{min} \leq u_{k+i} \leq u_i^{max} \tag{5.13d}$$

$$i = 0, \ldots, N_p - 1 \tag{5.13e}$$

3.3 is the solution of the problem for an unconstrained case if no violation of the constraints. The solver starts with the unconstrained case and searches if there is a violation. If exists, it solves the constraint optimization to produce the optimal control input under constraint. For the gimbal platform application, the control input is the only variable that we bound. The speed loop is responsible for providing the reference input for the torque loop. We take the limit on the torque so that MPC respects the limitation on torque when generating optimal input. According to system dynamics, the torque has a direct relation with acceleration, i.e., $\frac{T_e}{J} = \dot{w}$. Thus, the rate of change of the desired set-point for the speed loop will not be limited. We formulate the torque constraint with the following equation.

$$u \in \mathbb{R} \text{ s.t. } |u| < T_{max} \tag{5.14}$$

5.3 Results

We verified the effectiveness of the MPC over classical methods via experimental results on the real gimbal platform. We compare the results of MPC with standard PID controller to demonstrate the improvements in the tracking performance of the speed controller. We implement MPC on the custom-made controller unit that includes the C6713B processor. The test bench depicted in Fig.5.1 includes a gimbal platform that is mounted on a fixed platform. The gimbal consists of the following components: direct-drive PMSM motor, custom-made controller unit, mechanical structure, high-resolution camera(for pointing and tracking), sub-components for assembly. Our feedback path includes the following sensors: encoder(for position and commutation), gyroscope(for angular speed and stabilization), and current transducers(for inner loop). The parameters of the system and controller are listed in Table. 5.1. The delay in the system is approximately $4.8ms$, which is almost equal to $T_d \approx 5T_s$. Thus, we extend the state-space model with five extra delay input resulting in 7 states in the overall system. We perform several simulations before implement-

Table 5.1: GIMBAL and CONTROLLER Parameters

Parameter	Value
J	$2.58kgm^2$
(T_s, T_c, T_w)	$(2.7Nm, 2.27Nm, 0.015\frac{Nm}{deg/s})$
(w_b, w_s)	$(0.3, 0.5)deg/s$
T_s	$1ms$
(H_p, H_u)	$(10, 5)$
(Q, R)	$(\mathbf{I}_{10\times10}, 0.01\mathbf{I}_{5\times5})$
(P, I)	$(0.25, 5)$
T_{max}	$26Nm$

ing the MPC on the real gimbal platform. The controller weighting matrices and other parameters such as prediction and control horizon are tuned to their final values via simulations. We also extend the prediction and control horizons in simulations which

do not enhance the tracking performance noticeably. Our test procedure includes two cases: first, improve the closed-loop performance at lower speed by eliminating the effect of friction. For this, we apply a sinusoidal wave and observe the zero-crossing point of our implementation. As the second case, we apply the step-like set-point for the speed loop to operate the system close to the limits. We compare the MPC with standard PID to demonstrate the improvements in the tracking. As for the first case, the MPC usually generates the optimal control input from an unconstrained solution since the rate change of the speed loop is generally less than the upper bound of the acceleration with sinusoidal input. On the other hand, the optimization solves the constrained optimization problem for step-like input that generates the optimal control signal while respecting torque constraint. Fig. 5.6 illustrates the improvements of the MPC for the friction effect by comparing standard PID.

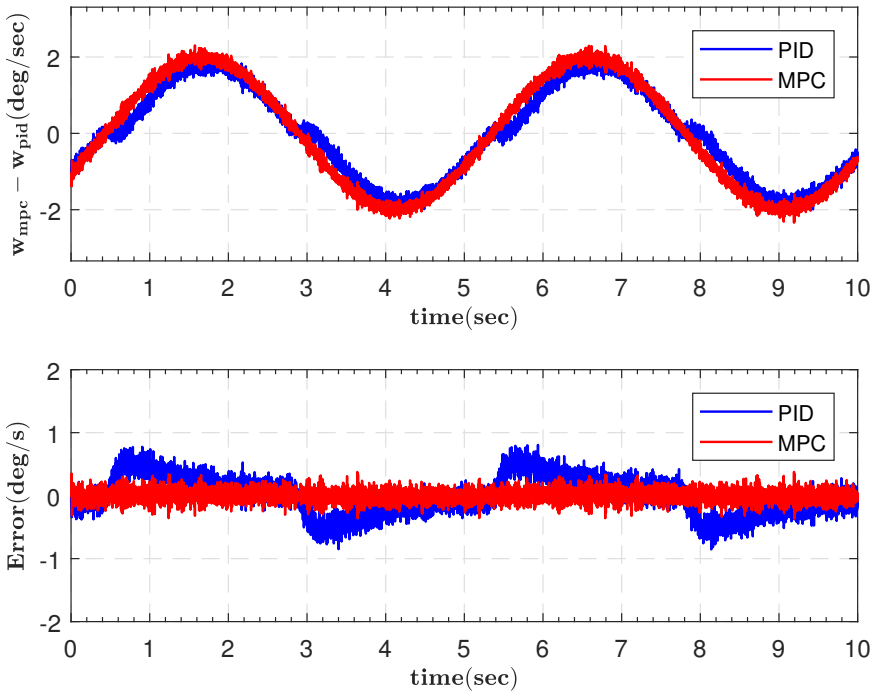

Figure 5.6: The improvement with MPC on the friction effect that causes to stick the gimbal platform at zero crossing.

In the second example, we apply the step-line set-point for the speed loop as given in Fig. 5.7. The PID controller does not include any add-ons that explicitly enable us to compare the two controllers' performances. We observe the effect of the overshoot that takes place in PID because of integral wind-up phenomena. The MPC is aware

of the constraints during the prediction and control horizon that allows taking action for producing the optimal control input to the system so that the output has almost no overshoot, whereas PID has. We compare the effectiveness of the two algorithms according to ISE(integral-square of error) criteria. There is an approximately 4.8% improvement in MPC.

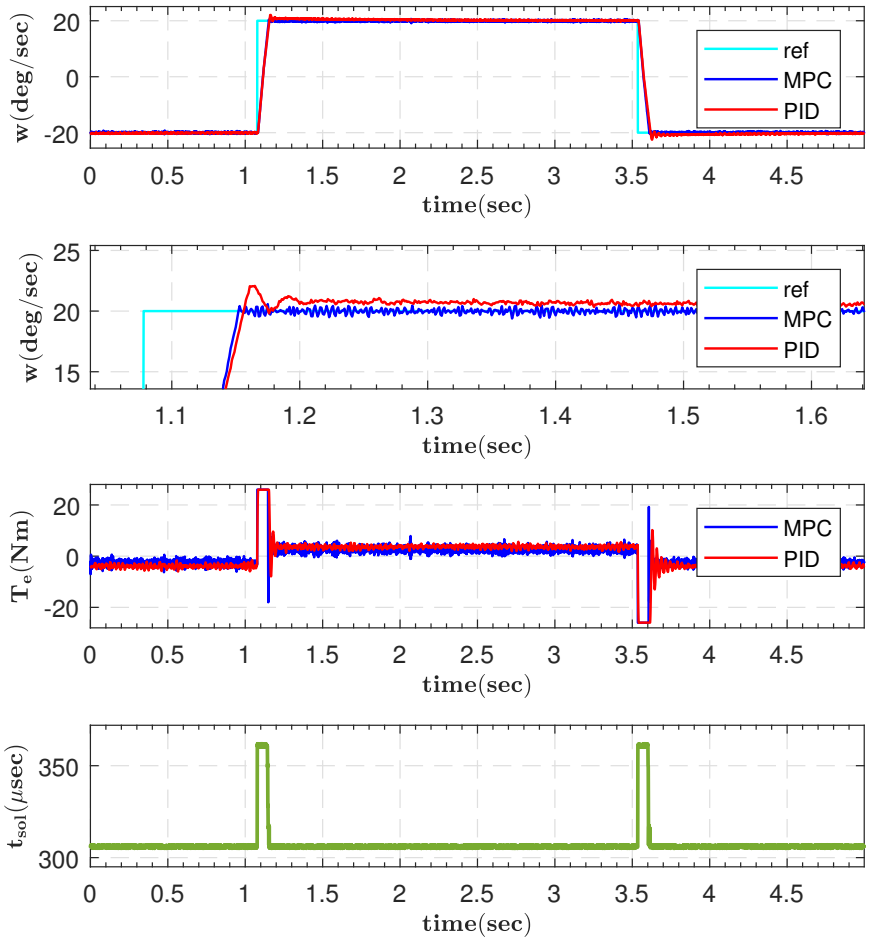

Figure 5.7: The improvement with MPC for square wave input reference under torque limitation.

CHAPTER 6

CONCLUSION & FUTURE WORKS

This thesis demonstrated the feasibility of online MPC with different industrial applications by applying the beneficial properties of linear MPC over classical control strategies. The dual active set solver with an efficient matrix updating procedure has been successfully carried out to find the optimal control signals for the system by satisfying the system constraints. We evaluate the proposed approach in two different literature examples to demonstrate its performance by comparing it with literature results.

As the first industrial application, we chose to perform the MPC in PMSM with field weakening operation. We evaluate the performances of the algorithm in both constant torque and constant power regions. To operate the motor in a constant power region, i.e., field weakening region, we include voltage limitation inside the algorithm to generate proper d axis current reference. The reason to perform the proposed algorithm in two different zones is to demonstrate the feasibility of our implementation under direct torque control operation. We have rearranged the linear approximation on the constraints by combining the two constraint lines to reduce the number of constraints. The feasibility of the proposed controller structure was verified via PIL simulation and physical experiment. The results demonstrated the practical feasibility of the algorithm to control the PMSM in the laboratory environment.

We chose to control the one axis of the gimbal platform as a second example. The well-known friction model is re-arranged as the combination of linear lines so as to include it in the linear MPC. We compare the results with the classical approach to eliminate the effect of friction where it affects the system's performance, especially at the lower speed. Furthermore, the proposed approach was verified in the low-cost

motion control unit to certify that our implementation is viable. As a further contribution to the gimbal platform, we will investigate the stabilization of the gimbal platform under known and unknown disturbances. We will include the measured disturbances in the model and add integral action for unknown disturbances to increase the effectiveness of the algorithm under disturbances.

Since the model losses the information with linear assumptions, the linearized model may sometimes not provide sufficient information about the overall dynamics of the system to the model-based controller. Therefore, it is necessary to capture the nonlinear characteristic of the system where the linearized is not sufficient to increase the quality of the controller. In this context, our research will continue to extend the model-based approach control structure for Linear Time-Varying(LTV) and Linear Parameter- Varying(LPV) models, which are suitable alternatives to represent the nonlinear dynamics of the system, to increase the performances.

REFERENCES

[1] P. D. Vouzis, L. G. Bleris, M. G. Arnold, and M. V. Kothare, "A system-on-a-chip implementation for embedded real-time model predictive control," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 1006–1017, 2009.

[2] D. Y. Ohm, "Dynamic model of pm synchronous motors," *Drivetech, Inc., Blacksburg, Virginia, www. drivetechinc. com*, vol. 16, 2000.

[3] S. Kouro, M. A. Perez, J. Rodriguez, A. M. Llor, and H. A. Young, "Model predictive control: MPC's role in the evolution of power electronics," *IEEE Industrial Electronics Magazine*, vol. 9, no. 4, pp. 8–21, 2015.

[4] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex Optimization*. Cambridge university press, 2004.

[5] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[6] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 11, no. 7, pp. 733–764, 2003.

[7] J. M. Maciejowski, *Predictive control: With constraints*. Pearson education, 2002.

[8] D. W. Clarke, C. Mohtadi, and P. Tuffs, "Generalized predictive control—part i. the basic algorithm," *Automatica*, vol. 23, no. 2, pp. 137–148, 1987.

[9] C. R. Cutler and B. L. Ramaker, "Dynamic matrix control - a computer control algorithm," in *Joint Automatic Control Conference*, no. 17, p. 72, 1980.

[10] P. Cortés, M. P. Kazmierkowski, R. M. Kennel, D. E. Quevedo, and J. Rodríguez, "Predictive control in power electronics and drives," *IEEE Transactions on Industrial Electronics*, vol. 55, no. 12, pp. 4312–4324, 2008.

[11] S. Vazquez, J. I. Leon, L. G. Franquelo, J. Rodriguez, H. A. Young, A. Marquez, and P. Zanchetta, "Model predictive control: A review of its applications in power electronics," *IEEE Industrial Electronics Magazine*, vol. 8, no. 1, pp. 16–31, 2014.

[12] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[13] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit solution of model predictive control via multiparametric quadratic programming," in *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No. 00CH36334)*, vol. 2, pp. 872–876. IEEE, 2000.

[14] A. G. Wills, G. Knagge, and B. Ninness, "Fast linear model predictive control via custom integrated circuit architecture," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 1, pp. 59–71, 2011.

[15] G. Cimini, D. Bernardini, S. Levijoki, and A. Bemporad, "Embedded model predictive control with certified real-time optimization for synchronous motors," *IEEE Transactions on Control Systems Technology*, 2020.

[16] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[17] S. a. Keerthi and E. G. Gilbert, "Optimal infinite-horizon feedback laws for a general class of constrained discrete-time systems: Stability and moving-horizon approximations," *Journal of Optimization Theory and Applications*, vol. 57, no. 2, pp. 265–293, 1988.

[18] J. Rawlings and K. Muske, "The stability of constrained receding horizon control," *IEEE Transactions on Automatic Control*, vol. 38, DOI 10.1109/9.241565, no. 10, pp. 1512–1516, 1993.

[19] S. Bolognani, S. Bolognani, L. Peretti, and M. Zigliotto, "Design and implementation of model predictive control for electrical motor drives," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 6, pp. 1925–1936, 2008.

[20] P. E. Gill, N. I. Gould, W. Murray, M. A. Saunders, and M. H. Wright, "A weighted gram-schmidt method for convex quadratic programming," *Mathematical Programming*, vol. 30, no. 2, pp. 176–195, 1984.

[21] A. Bemporad, "A quadratic programming algorithm based on nonnegative least squares with applications to embedded model predictive control," *IEEE Transactions on Automatic Control*, vol. 61, no. 4, pp. 1111–1116, 2015.

[22] A. Forsgren, P. E. Gill, and E. Wong, "Primal and dual active-set methods for convex quadratic programming," *Mathematical Programming*, vol. 159, no. 1, pp. 469–508, 2016.

[23] P. Giselsson and S. Boyd, "Metric selection in fast dual forward–backward splitting," *Automatica*, vol. 62, pp. 1–10, 2015.

[24] D. Goldfarb and A. Idnani, "A numerically stable dual method for solving strictly convex quadratic programs," *Mathematical Programming*, vol. 27, no. 1, pp. 1–33, 1983.

[25] R. A. Bartlett and L. T. Biegler, "Qpschur: a dual, active-set, schur-complement method for large-scale and structured convex quadratic programming," *Optimization and Engineering*, vol. 7, no. 1, pp. 5–32, 2006.

[26] G. H. Golub and C. F. Van Loan, *Matrix Computations*, vol. 3. JHU press, 2013.

[27] G. Knagge, A. Wills, A. Mills, and B. Ninness, "Asic and fpga implementation strategies for model predictive control," in *2009 European Control Conference (ECC)*, pp. 144–149. IEEE, 2009.

[28] N. J. Higham, *Accuracy and stability of numerical algorithms*. SIAM, 2002.

[29] TEXAS INSTRUMENT, "C28x Floating Point Unit fastRTS Library Module: User's Guide," 2010.

[30] TEXAS INSTRUMENT, "TMS320C28x FPU Primer, SPRAAN9A," 2009. [Online]. Available: https://www.ti.com/lit/an/spraan9a/spraan9a.pdf

[31] J. Currie, "Practical applications of industrial optimization: from high-speed embedded controllers to large discrete utility systems: a thesis submitted to

auckland university of technology in fulfilment of the requirements for the degree of doctor of philosophy (phd), 2014," Ph.D. dissertation.

[32] A. K. Abbes, F. Bouani, and M. Ksouri, "A microcontroller implementation of constrained model predictive control," *World Academy of Science, Engineering and Technology*, vol. 5, no. 8, pp. 655–662, 2011.

[33] K.-V. Ling, B. F. Wu, and J. Maciejowski, "Embedded model predictive control (mpc) using a fpga," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 15 250–15 255, 2008.

[34] M. S. Lau, S.-P. Yue, K. V. Ling, and J. M. Maciejowski, "A comparison of interior point and active set methods for fpga implementation of model predictive control," in *2009 European Control Conference (ECC)*, pp. 156–161. IEEE, 2009.

[35] M. V. Kothare, V. Balakrishnan, and M. Morari, "Robust constrained model predictive control using linear matrix inequalities," *Automatica*, vol. 32, no. 10, pp. 1361–1379, 1996.

[36] M. Preindl and S. Bolognani, "Model predictive direct torque control with finite control set for pmsm drive systems, part 1: Maximum torque per ampere operation," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 4, pp. 1912–1921, 2013.

[37] M. Preindl and S. Bolognani, "Model predictive direct torque control with finite control set for pmsm drive systems, part 2: field weakening operation," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 2, pp. 648–657, 2012.

[38] Z. Mynar, L. Vesely, and P. Vaclavek, "Pmsm model predictive control with field-weakening implementation," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 8, pp. 5156–5166, 2016.

[39] J. Liu, C. Gong, Z. Han, and H. Yu, "Ipmsm model predictive control in flux-weakening operation using an improved algorithm," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 12, pp. 9378–9387, 2018.

[40] Y. Zhang, B. Zhang, H. Yang, M. Norambuena, and J. Rodriguez, "Generalized sequential model predictive control of im drives with field-weakening ability," *IEEE Transactions on Power Electronics*, vol. 34, no. 9, pp. 8944–8955, 2018.

[41] Z. Zheng and D. Sun, "Model predictive flux control with cost function-based field weakening strategy for permanent magnet synchronous motor," *IEEE Transactions on Power Electronics*, vol. 35, no. 2, pp. 2151–2159, 2019.

[42] J. Su, R. Gao, and I. Husain, "Model predictive control based field-weakening strategy for traction ev used induction motor," *IEEE Transactions on Industry Applications*, vol. 54, no. 3, pp. 2295–2305, 2017.

[43] S. Chai, L. Wang, and E. Rogers, "A cascade mpc control structure for a pmsm with speed ripple minimization," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 8, pp. 2978–2987, 2012.

[44] M. Novak, J. Novak, and J. Chysky, "Experimental verification of high-speed permanent magnet synchronous motor model," in *2012 XXth International Conference on Electrical Machines*, pp. 2435–2440. IEEE, 2012.

[45] M. Kazerooni and N. C. Kar, "Methods for determining the parameters and characteristics of pmsm," in *2011 IEEE International Electric Machines & Drives Conference (IEMDC)*, pp. 955–960. IEEE, 2011.

[46] M. Preindl, S. Bolognani, and C. Danielson, "Model predictive torque control with pwm using fast gradient method," in *2013 Twenty-Eighth Annual IEEE Applied Power Electronics Conference and Exposition (APEC)*, pp. 2590–2597. IEEE, 2013.

[47] J. Hilkert, "Inertially stabilized platform technology concepts and principles," *IEEE Control Systems Magazine*, vol. 28, no. 1, pp. 26–46, 2008.

[48] M. K. Masten, "Inertially stabilized platforms for optical imaging systems," *IEEE Control Systems Magazine*, vol. 28, no. 1, pp. 47–64, 2008.

[49] H. Olsson, K. J. Åström, C. C. De Wit, M. Gäfvert, and P. Lischinsky, "Friction models and friction compensation," *Eur. J. Control*, vol. 4, no. 3, pp. 176–195, 1998.

[50] J. Swevers, F. Al-Bender, C. G. Ganseman, and T. Projogo, "An integrated friction model structure with improved presliding behavior for accurate friction compensation," *IEEE Transactions on Automatic Control*, vol. 45, no. 4, pp. 675–686, 2000.

[51] C. C. De Wit, H. Olsson, K. J. Astrom, and P. Lischinsky, "A new model for control of systems with friction," *IEEE Transactions on Automatic Control*, vol. 40, no. 3, pp. 419–425, 1995.

[52] L. Márton and B. Lantos, "Control of mechanical systems with stribeck friction and backlash," *Systems & Control Letters*, vol. 58, no. 2, pp. 141–147, 2009.

[53] M. Jankovic and I. Kolmanovsky, "Developments in control of time-delay systems for automotive powertrain applications," in *Delay Differential Equations*, pp. 55–92. Springer, 2009.

[54] S. Di Cairano, D. Yanakiev, A. Bemporad, I. V. Kolmanovsky, and D. Hrovat, "Model predictive idle speed control: Design, analysis, and experimental evaluation," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 1, pp. 84–97, 2011.

APPENDIX A

MATRIX UPDATING PART IN SOLVER

Two common approaches to compute QR decomposition, i.e., Givens Rotation and Householder Reflection that we utilize to update the matrices in the solver are provided with basic examples in this section of the thesis.

A.1 Givens Rotations

Givens Rotation(or plane rotation) introduces zeros to only one element of the vector in each calculation. To be more precise, Given Rotation of a vector $[\alpha\ \beta]^T$ requires an orthogonal rotation matrix, which is

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix} \tag{A.1}$$

where $r = \sqrt{\alpha^2 + \beta^2}$. The elements in the rotation matrix are calculated as,

$$c = \alpha / r \tag{A.2}$$

$$s = -\beta / r \tag{A.3}$$

An example to compute the QR decomposition of matrix $A \in \mathbb{R}^{3\times 3}$ via Givens Rotation method is provided in the following,

$$A = \begin{bmatrix} X & X & X \\ X & X & X \\ X & X & X \end{bmatrix} \overrightarrow{G_1} \begin{bmatrix} X & X & X \\ X & X & X \\ 0 & X & X \end{bmatrix} \overrightarrow{G_2} \left[\begin{array}{c|cc} X & X & X \\ \hline 0 & X & X \\ 0 & X & X \end{array}\right] \overrightarrow{G_3} \left[\begin{array}{c|cc} X & X & X \\ \hline 0 & X & X \\ 0 & 0 & X \end{array}\right] \tag{A.4}$$

Hence, we obtain

$$\underbrace{G_3 G_2 G_1}_{\tilde{Q}} A \iff A = QR \tag{A.5}$$

where $\tilde{Q} = Q^T$.

To be more precise, a numerical example of 2×2 matrix is provided in order to compute the QR decomposition via Givens Rotation method.

$$A = \begin{bmatrix} 3 & 1 \\ 4 & 1 \end{bmatrix} \tag{A.6}$$

The rotation matrix is calculated according to equations in (A.1,A.2) for the first column of the A matrix,

$$G_1 = \begin{bmatrix} 0.6 & 0.8 \\ -0.8 & 0.6 \end{bmatrix} \tag{A.7}$$

G_1 is applied to A to compute the QR decomposition as,

$$\underbrace{\begin{bmatrix} 3 & 1 \\ 4 & 1 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 0.6 & -0.8 \\ 0.8 & 0.6 \end{bmatrix}}_{Q} \underbrace{\begin{bmatrix} 5 & 1.4 \\ 0 & -0.2 \end{bmatrix}}_{R} \tag{A.8}$$

A.2 Householder Reflection

The matrix updating procedure in constraint addition via Householder Reflection method is provided in this section with a basic example for the ease of the reader.

The reflection matrix is computed with the following equations,

$$H = I - 2vv^T \tag{A.9}$$

The application of the reflection matrix H to a vector,

$$Hx = (I - \frac{2uu^T}{u^T u})x \tag{A.10}$$

where

$$u = \frac{v}{\|v\|} \tag{A.11}$$

$$u = \begin{bmatrix} x_1 + sign(x_{11}) \|x_1\| \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{A.12}$$

We will apply the same example that we use in Givens Rotation section. In order to build the reflection matrix, u vector is computed as

$$\|x_1\| = \sqrt{3^2 + 4^2} = 5 \tag{A.13}$$

$$u = x_1 + sign(x_{11}) \|x_1\| e_1 = \begin{bmatrix} 3 \\ 4 \end{bmatrix} + 5 \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 8 \\ 4 \end{bmatrix} \tag{A.14}$$

By using the equation in (A.10), the reflection matrix is computed as,

$$H_1 = I - \frac{2uu^T}{u^T u} = \begin{bmatrix} -0.6 & -0.8 \\ -0.8 & 0.6 \end{bmatrix} \tag{A.15}$$

H_1 is applied to A to compute the QR decomposition as,

$$\underbrace{\begin{bmatrix} 3 & 1 \\ 4 & 1 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} -0.6 & -0.8 \\ -0.8 & 0.6 \end{bmatrix}}_{Q} \underbrace{\begin{bmatrix} -5 & -1.4 \\ 0 & -0.2 \end{bmatrix}}_{R} \tag{A.16}$$