Low-Complexity Optimization-Based Control: Design, Methods and Applications

Thèse N° 9564

Présentée le 25 octobre 2019

à la Faculté des sciences et techniques de l'ingénieur
Laboratoire d'automatique 3
Programme doctoral en génie électrique

pour l'obtention du grade de Docteur ès Sciences

par

Ivan PEJCIC

Acceptée sur proposition du jury

Dr A. Karimi, président du jury
Prof. C. N. Jones, directeur de thèse
Prof. P. Giselsson, rapporteur
Prof. T. Besselmann, rapporteur
Prof. D. Kuhn, rapporteur

2019

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

To my parents and sister

Acknowledgements

This first paragraph of the thesis I would like to use to express my sincere gratitude to my supervisor, Prof. Colin Jones. The first time I encountered Colin was as a master student attending his course at EPFL where I was able to notice his genuine interest in science and his special relation towards the course subject, which he was teaching with fluency that only appears after a great amount of effort and dedication. I am glad that my doctoral studies allowed me to spend some time with a person like Colin and get influenced by his everyday attitude and way of thinking. I would like to thank Colin for the great time we had working together during my doctoral studies, for providing me an opportunity to enter into a field of science which I consider interesting and important and, above all, I would like to thank him for the personal development which feels great and which collaboration with him provided. I wish him and his lab all the best for the future scientific career, and I am looking forward for occasions in the future involving technical discussions about interesting scientific ideas.

The environment in which I spent my doctoral studies allowed me to encounter many interesting and friendly people. I would like to thank other automatic control professors at EPFL, Dominique Bonvin, Alireza Karimi, Giancarlo Ferrari Trecate, Christophe Salzmann and Philippe Müllhaupt for bringing great people to the laboratory and for always being open for providing advices and help. The administrative support of the secretaries in the laboratory, Ruth, Eva, Margot and Nicole is also greatly appreciated.

I would like to thank Stefan Almér and Helfried Peyrl from ABB Switzerland for a fruitful collaboration that we had together, as well as my colleagues Altug Bitlislioglu, Milan Korda and Harsh Shukla from EPFL for collaborating on papers that we coauthored. I would also like to use this opportunity to thank Peter Hokayem from ABB for a pleasant and productive collaboration that we had during my internship and master thesis in ABB.

A special thanks goes to all my colleagues and friends from EPFL for enriching my studies. An attempt to mention some of their names would be a list involving Altug, Luca, Francisco, Tomasz, Andrea, Jean, Milan, Ye, Faran, Georgios, Sanket, Yingzhao, Petr, Emilio, Harsh, Ioannis, Predrag, Marko, Diogo, Zlatko, Sean, Shriniketh, Martand, Rene, Timm, Tafarel, Christoph, Mahdieh, Sohail, Achille, Michele, Mustafa and Pulkit.

Finally, I would like to thank my parents for rising me to become a combination of their characters and for their support without which this thesis would not be possible, and I would also like to specially thank my sister for providing good advices and perspective when they were needed, as well as for having a very similar sense of humor to mine.

Lausanne, June 2019 I. P.

Abstract

Optimization-based controllers are advanced control systems whose mechanism of determining control inputs requires the solution of a mathematical optimization problem. In this thesis, several contributions related to the computational effort required for optimization-based controller execution are provided. The content of the thesis is divided into three parts:

The first part provides methods capable of performing automatic controller tuning for constrained control of nonlinear systems. Given a specified controller structure, the presented methods are able to perform an offline tuning of the controller parameters such that some user-specified performance metric is optimized while imposing stability guarantees on the obtained closed-loop system. The methods are characterized by a broad flexibility that allows their application to many control schemes that are widely popular in practice, but also to novel user-specified control schemes that are convenient from a computational or some other point of view. The controller tuning is formulated as an optimization problem that can be tackled by black-box optimization techniques such as Bayesian optimization. The methods are demonstrated by application examples involving speed control of a permanent magnet synchronous machine and position control of a mechanical gyroscopic system.

The second part provides an accelerated version of the alternating direction method of multipliers (ADMM) optimization algorithm derived by using a recently proposed accelerated Douglas-Rachford (DR) splitting. The obtained method is an accelerated ADMM version that replaces the internal proximal point convergence mechanism of the classical ADMM by the accelerated gradient method applied on a specially constructed scaled DR envelope function. The form of the accelerated ADMM is derived and conditions are provided under which the underlying accelerated DR splitting is validly addressing the Fenchel dual problem.

The third part describes a model predictive control scheme for power electronics control which involves a combination of the integral of squared predicted tracking error as the controller's cost function together with offline computed optimal steady-state voltage signals. These offline computed optimal steady-state signals are in the power electronics community referred to as Optimized Pulse Patterns (OPPs). The method is presented by considering an industrial case study involving a grid-tied converter with LC filter. After introducing an optimal control problem based on OPPs, low computational complexity approximate versions are provided. The resulting approximate controller versions are addressed by using memory storage of the dynamic behavior of the system, leading to controller forms whose execution can be performed on embedded hardware.

Abstract

Keywords: optimization-based control, MPC, controller tuning, nonlinear systems, sum-of-squares, scenario approach, Bayesian optimization, ADMM, Douglas-Rachford splitting, Nesterov acceleration, power electronics, optimized pulse patterns, multilevel converters

Résumé

Les régulateurs basés sur l'optimisation sont des régulateurs avancés dont le mécanisme de détermination de l'entrée repose sur la solution d'un problème d'optimisation mathématique. Dans cette thèse, plusieurs contributions en relation sur le coût calculatoire de l'implémentation du régulateur basé sur l'optimisation résultant, seront présentées. Le contenu de cette thèse est divisée en trois parties.

La première partie traite des méthodes capables d'ajuster de manière automatique les paramètres du règulateur et s'applique aux systèmes non-linéaires contraints. A partir d'une structure de régualteur donnée, la méthode présentée est capable d'effectuer un ajustage hors-ligne des paramètres du régulateur en garantissant à la fois une performance utilisateur donnée et une stabilité de la boucle fermée. Les méthodes proposées ont un caractère étendu et flexible rendant possible leur application à une vaste palette de problèmes d'automatique rencontré dans la pratique, mais également rend possible des innovations de techniques existantes en améliorant les performances calculatoires et d'autres indices de performance. L'ajustage du régulateur est formulé comme un problème d'optimisation qui peut être résolu comme un problème d'optimisation boîte noire telle que l'optimisation Bayesienne. Les méthods sont validées à travers des exemples d'application telles que la régulation de vitesse d'un moteur à aimants permanents (machine synchrone) et le positionnement d'un système gyroscopique mécanique.

La seconde partie propose une méthode pour accélérer la technique d'optimisation fondées sur les directions alternées dans la méthode des mulitiplicateurs (ADMM). Elle découle d'une méthode d'accélération récente appelée méthode par scindement Douglas-Rachford (DR). La méthode ainsi proposée est est un version accélérée ADMM qui remplace le mécanismae de convergence ponctuel par méthode interne proximale de la méthode classique ADMM par une méthode de grandients accélérée appliquée à une fonction enveloppe construite spécialement et mise en échelle selon le critère Douglas-Rachford. La forme issue d'une telle modification de la méthode ADMM est ainsi obtenue et caractérisée de telle sorte à obtenir les conditions pour lesquelles le scindement DR est possible tout en adressant le problème de Fenchel dual.

La troisième partie décrit un schéma de commande prédictive fondée sur un modèle avec comme application la régulation de systèmes d'électronique de puissance. Elle comporte une fonction de coût de commande qui repose sur un terme qui évalue l'intégrale du carré de l'erreur de prédiction de poursuite combinée avec un terme qui évalue la valeur optimale hors

ligne des niveaux asymptotiques des tensions des signaux. Ces valeurs des signaux asymptotiques hors-lignes sont référés dans la litérature de l'électronique de puissance comme les modèles d'impulsions optimisés (OPPs, Optimized Pulse Patterns). La méthode présentée est appliquée à un cas d'étude industriel qui repose sur un convertisseur LC attaché au réseau. Après avoir appliqué un problème de commande optimale basée sur les OPPs, des approximations de complexité calculatoire réduite sont proposées. Les régulateurs approximatifs ainsi obtenus sont implémentés en tirant parti du stockage en mémoire du comportement dynamique du système, rendant ainsi possible une structure implémentable sur des systèmes embarqués.

Mots-clés : Commande basée sur l'optimistaion, MPC, ajustage de régulateur, systèmes non linéaires, somme de carrés, approche scénario, optimisation bayésienne, ADMM, scindement Douglas-Rachford, accélération de Nestorov, électronique de puissance, modèles d'impulsions optimisés (OPPs), convertisseurs multi-niveaux.

Contents

List of Figures

List of Tables

1 Introduction

Optimization-based controllers are advanced control systems whose mechanism of determining control inputs requires the solution of a mathematical optimization problem. Controllers of this kind are usually formed by using a prediction horizon concept; that is, over a certain prediction horizon into the future, a system model is used to predict the future evolution of the system as a function of the predicted inputs, and by usage of prediction the best input signal over the prediction horizon is determined so that some cost function is minimized. To provide feedback and robustness, after computing the best control input over the prediction horizon, only its portion over the first control period is applied and the whole computational process is repeated at the next control period by using new information about the system state. In comparison to classical control schemes based on linear systems theory, optimization-based controllers are characterized by important advantages such as direct treatment of the system's constraints and ability to treat nonlinear system dynamics, resulting thereby in a better control performance.

Beside the ability to provide better control performance, optimization-based control also involves several drawbacks. One of the most prominent is the computational effort required for execution of the control algorithm. This usually high computational effort is caused by a necessity to solve the involved optimization problem to compute the control input. An additional difficulty is caused by a necessity to select a convenient optimization algorithm and appropriately tune it for the present control problem. The selection of an optimization algorithm should consider several aspects, such as exploitation of the optimization problem's structure, robustness to numerical errors caused by finite precision of the computational hardware, as well as the convergence properties whose establishing usually requires a mathematically rigorous analysis. Despite the rich theoretical foundations of control theory and mathematical optimization, a practical implementation of optimization-based control often requires a considerable amount of trial-and-error parameter tuning of the cost function and the optimization algorithm in order to achieve stability and the optimization algorithm's practical convergence rate appropriate for real-time execution within the available control period.

This thesis provides several contributions related to to the computational effort required for optimization-based controller execution. The content of the thesis is divided into three parts, as sumamrized in what follows.

Part I - Automatic Controller Tuning

The first part of the thesis presents methods capable of performing offline automatic tuning of controller parameters. More specifically, given some fixed controller structure, the methods of this part tune the controller parameters such that some user-specified performance metric is optimized while imposing stability guarantees on the obtained closed-loop system. The techniques are characterized by a broad flexibility that allows their application to nonlinear system dynamics, user-specified controller structures and problem-tailored performance metrics. As such, they allow the design of low-complexity optimization-based controllers whose tuning parameters (including possibly also those of the optimization algorithm) are optimized for performance and closed-loop stability. Due to the high flexibility in terms of the controller structures and performance metrics that can be considered, the presented tuning methods allow not just an easier design of optimization-based controllers by tuning their control and optimization algorithm parameters, but also provide the user with a tool capable of tuning his intuitive novel control structures in order to bring them to their best performance while imposing closed-loop stability. The content consists of the three chapters described below.

Chapter 2 - Automatic tuning based on sum-of-squares programming

This chapter describes a tuning technique which involves sum-of-squares (SOS) programming for the generation of closed-loop Lyapunov functions for polynomial systems. The SOS stability verification technique used in this chapter is the one from [43]. In comparison to other controller design methods using SOS programming, the method of this chapter is characterized by the ability to address nonlinear polynomial dynamics in a constrained setting with broad flexibility regarding the performance metrics which can be specified by the user. To provide an illustration of the possibilities which the tuning method creates in case of model predictive control (MPC), an example involving synthesis of an explicit MPC (EMPC) controller for constrained control of nonlinear system dynamics is considered. As will be described in more detail in the chapter, the tuning method consists of two phases, both of which can be addressed by using a black-box optimization technique such as Bayesian optimization which will be used in the numerical example.

The content of this chapter is based on the following publication:

- I. Pejcic, M. Korda, C. N. Jones. "Control of Nonlinear Systems with Explicit-MPC-like Controllers", In Proc. of Conference on Decision and Control, 2017.

whose content has also partly appeared in:

- M. Kvasnica, C. N. Jones, I. Pejcic, J. Holaza, M. Korda, P. Bakarac. "Real-Time Implementation of Explicit Model Predictive Control", In Handbook of Model Predictive Control, Springer, 2018.

Chapter 3 - Extension to multimodel uncertainty and experimental verification

This chapter brings several further developments related to the automatic tuning method of Chapter 2. Besides demonstrating a broader applicability of the tuning method from Chapter 2 by applying it in a case involving a non-optimization-based, nonlinear control policy, the primary purpose of this chapter is to experimentally demonstrate the validity of the method by application to a mechanical physical system and to extend the method's practical computational capability to the case of multimodel plant uncertainty. The mentioned broader generality of the method from Chapter 2 (i.e., applicability to non-optimization-based control schemes) can be anticipated from some of the illustrative examples provided in papers [43] and [42] from which the SOS stability certification used in Chapter 2 and this chapter originates. The anti-windup equipped PID control scheme considered in this chapter is embedded into the SOS framework by using the KKT conditions in a manner similar to an example presented in [43].

The content of this chapter is based on the publication:

- I. Pejcic and C. N. Jones. "Experimental Verification of Sum-Of-Squares-Based Controller Tuning Technique with Extension to Parallel Multimodel Uncertainty Processing", Accepted to European Control Conference 2019.

Chapter 4 - Automatic tuning based on scenario approach technique

This chapter presents a method whose purpose is to extend the benefits of the automatic tuning procedure described in Chapter 2 to problems of larger size. The presented method also provides better modelling flexibility than the SOS approach as it does not require polynomial form for the functions describing the system. Instead of using the SOS programming technique for computation of Lyapunov functions, a procedure allowing a high accuracy numerical Lyapunov function estimation is introduced. This procedure formulates the Lyapunov function search as a robust optimization problem which is then tackled by applying a Chebychev center and scenario-based optimization techniques. The concept is then incorporated into a controller synthesis method whose demonstration is performed by application to two examples which cannot be addressed by the SOS-based approach due to scalability and modelling limitations.

This chapter is based on a publication in preparation:

- I. Pejcic and C. N. Jones. "A General Automatic Tuning Method for Constrained Nonlinear Systems Control", In preparation.

Part II - Accelerated ADMM based on Accelerated Douglas-Rachford Splitting

Chapter 5 - Accelerated ADMM based on Accelerated Douglas-Rachford Splitting

Solving of the controller's optimization problem can be addressed by a large variety of optimization algorithms. An algorithm that has received attention in recent years is the alternating direction method of multipliers (ADMM). A general property of first order methods, including the gradient method and ADMM, is their slow convergence rate in comparison to advanced second order methods. A substantial contribution to the performance of the gradient method is achieved through its acceleration based on an extrapolation rule between subsequent gradient steps. This chapter provides an accelerated version of ADMM by building on a recently proposed accelerated Douglas-Rachford (DR) splitting, resulting in a method that replaces the internal proximal point convergence mechanism of classical ADMM by the accelerated gradient method applied on a specially constructed scaled DR envelope function. The chapter derives the form of the accelerated ADMM algorithm and provides conditions under which the underlying accelerated DR splitting is valid.

The content of this chapter is based on the publication:

- I. Pejcic and C. N. Jones. "Accelerated ADMM based on Accelerated Douglas-Rachford Splitting", In Proc. of European Control Conference, 2016.

Part III - Power Converter Pulse Pattern Optimal Control

Chapter 6 - Power Converter Pulse Pattern Optimal Control

The field of medium-voltage power electronics has demonstrated itself as a fruitful ground for application of optimization-based control. In comparison to the traditional control approaches that exist in the field, the developed optimization-based schemes showed an ability to get the existing power electronics hardware closer to its full operating potential, providing better efficiency and dynamic/steady-state performance. The method to be presented in this chapter is based on usage of offline computed optimal steady-state input signals that are in the power electronics community known as Optimized Pulse Patterns (OPPs). In comparison to the other methods which use OPPs to store the optimal steady-state operation in the controller's memory, the method of this chapter also allows a usage of memory to store the information about dynamic system behavior, thereby allowing an approximate low-computational complexity MPC that optimizes the transient behavior of complex power

electronics configurations for which it is not possible to approximate the plant dynamics with two integrators, as done by the state-of-the-art method. The presented method is thus more general (e.g., directly applicable in the presence of *LC* filters) and furthermore, it also involves penalization of the tracking error not only at the end of the prediction horizon but along it, thus not experiencing a degradation of transient performance when longer prediction horizons are used. Introduction of various problem-specific approximations allows a computational implementation on a field-programmable gate array (FPGA), and one particular FPGA model is considered in the numerical results section for assessment of the computational effort.

This chapter is mainly based on the publication:

- I. Pejcic, S. Almer, H. Peyrl. "Voltage Source Converter MPC with Optimized Pulse Patterns and Minimization of Integrated Squared Tracking Error". In Proc. of American Control Conference, 2017.

as well as on a publication in preparation:

- I. Pejcic, H. Shukla, S. Almer, J. Ferreau, C.N. Jones, "Low Computational Complexity Power Converter Optimal Control using Optimized Pulse Patterns", In preparation.

Part I

Automatic Controller Tuning

There is a large number of control schemes that are very popular from a practical point of view, but are not as theoretically sound as their closely related versions usually studied in the control literature. Employment of such control schemes often requires a considerable amount of effort spent on iterative trial-and-error tuning with an aim of eventually obtaining a controller that works satisfactorily, although without any closed-loop stability guarantees. Such circumstances create a need for an automatic tuning tool which should preferably be characterized by a generality that allows it to address a broad variety of user-specified control structures and closed-loop performance metrics.

The departures from theoretically sound controller forms are quite common in case of optimization-based controllers. In many cases, they are inevitable due to the practical limitations present in the implementation of the controller, causing for example a necessity for early termination of the controller's optimization algorithm due to hard real-time computational constraints, and resulting in a need for tuning of the optimization algorithm parameters in order to obtain an appropriate practical convergence rate. Another example involves model predictive control (MPC) schemes [68] used without a terminal constraint in order to obtain an optimization problem that is computationally easier to solve, causing thereby a loss of the stability properties guaranteed by the MPC theory. In the field of non-optimization-based controllers, an example are the controllers obtained by using linear systems theory to which saturation elements are subsequently added in order to handle system constraints, resulting as well in a loss of the performance/stability properties guaranteed by the linear systems theory and in a potentially considerable amount of effort to be spent on tuning by a technician.

This part of the thesis presents several methods capable of performing controller tuning for constrained control of nonlinear systems in an automated fashion. The techniques are characterized by a broad flexibility that allows their application to nonlinear system dynamics, user-specified controller structures and problem-tailored performance metrics. In particular, the presented techniques allow not only a tuning of many heuristic control schemes that are popular in practice, but also provide the user a possibility for development of his novel user-specified control schemes involving for example his intuitively designed early-terminated optimization algorithm (without convergence properties established in a mathematically rigorous fashion) whose parameters get tuned together with the other controller's parameters with respect to the given performance metric while imposing closed-loop stability on the obtained closed-loop system.

This Part I concerning automatic controller tuning consists of three chapters. The first of them, Chapter 2, develops an automatic controller tuning method that uses sum-of-squares (SOS) techniques for stability certification of the obtained closed-loop systems (i.e., for generation of closed-loop Lyapunov functions), involving thereby polynomial structure requirement on the functions describing the system. The method is demonstrated by synthesizing an explicit-MPC (EMPC) controller for speed control of a nonlinear permanent magnet synchronous machine (PMSM) model. The material of this chapter is based on the publication [61].

Chapter 3 describes extension of the SOS-based synthesis to the case of multimodel plant uncertainty, provides an experimental verification of the method by application on a mechanical gyroscopic system, and demonstrates applicability of the method beyond optimization-based controller structures by involving an anti-windup equipped PID controller synthesized robust to the multimodel uncertainty in the experimental setup. The material of this chapter is based on the document [60] accepted for publication.

Chapter 4 extends the benefits of the automatic tuning procedure presented in Chapter 2 to problems of larger size. The obtained method also provides better modelling flexibility than the SOS approach as it does not require polynomial form for the functions describing the system. Instead of using the SOS programming technique for computation of Lyapunov functions, a procedure allowing a high accuracy numerical Lyapunov function estimation is introduced. This procedure formulates the Lyapunov function search as a robust optimization problem which is then tackled by applying a Chebychev center and scenario-based optimization techniques. The concept is then incorporated into a controller synthesis method whose demonstration is performed by application to two examples which cannot be addressed by the SOS-based approach due to scalability and modelling limitations. The examples involve a synthesis of an early-terminated optimization-based controller applied for soft-constrained control of the PMSM rotational speed with the involved optimization algorithm tuned together with the other controller tuning parameters, as well as a synthesis of a cascaded linear controller with saturation applied for position control of gyroscope's disc, which is an input constrained nonlinear control problem.

2 Automatic tuning based on sum-of-squares programming

2.1 Introduction

This chapter describes a tuning technique which as one of its components uses sum-of-squares (SOS) programming for generation of closed-loop Lyapunov functions. The tuning method of this chapter will be based on the SOS stability verification technique from [43].

To provide an illustration of the possibilities which the tuning method creates, an example involving synthesis of explicit-MPC (EMPC) controller for control of nonlinear system dynamics is considered. EMPC represents a version of MPC [68] that stores the offline computed solution of the parametric Quadratic Program (QP) in the controller's memory for online usage, thus allowing one to avoid running an iterative optimization algorithm during controller's operation. One of the key restrictions of EMPC is the requirement to use a linear dynamic prediction model for EMPC formulation, which is necessary in order to ensure that a QP is obtained. In addition to it, due to the memory limitations EMPC often cannot involve long prediction horizons and is usually applied without terminal constraint, which is an element from MPC theory used to provide guarantees on the stability of the closed-loop system. An alternative approach to establish guaranteed closed-loop stability is an a-posteriori stability verification, which can be performed for a given controller after its design is finished, and which can be done in the case of a discrete-time linear system with QP controller by using the S-procedure [65] or Mixed Integer Linear Programing (MILP) [70]. In the case of a more general class which involves discrete-time systems described by polynomial functions, such a-posteriori stability verification can be done by using SOS programming [43].

The tuning allows synthesis of EMPC controllers with closed-loop stability guarantees for polynomial systems without relying on a terminal cost and/or constraint, but also even without using the prediction horizon concept (involving linear prediction model) to formulate the control optimization problem. In particular, for a specified QP structure the tuning method directly searches for the stabilizing coefficients in the cost and/or the constraints of the QP. This can be regarded as a search for a stabilizing control policy (i.e., a stabilizing mapping from measurements to plant inputs) which is described, in a compressed form, as a parametric QP

(a mapping from the QP input parameters to the optimal solution).

As will be described in what follows, the tuning method involves two phases. The first phase introduces a slack polynomial function to the stability verification technique [43] and allows a search for stabilizing tunings by a black-box optimization technique, such as Bayesian optimization which will be used in the numerical example of this chapter. The second phase takes the stabilizing tuning parameters from the first phase and optimizes for improvement of some user-specified performance criteria, as well by applying Bayesian optimization.

Controller synthesis using SOS techniques has been addressed in various forms in the existing studies. Paper [64] formulates synthesis of a stabilizing polynomial controller as a convex optimization problem, without considering closed-loop performance objective and system constraints. The development in [63] addresses nonlinear plant dynamics through a (non-unique) state-dependent linear system representation and synthesizes stabilizing polynomial controllers in unconstrained setting with minimized H_∞ or optimal cost (upper bounded by the obtained Lyapunov function) performance metrics, with success of the synthesis dependent on the choice of the non-unique state-dependent linear representation. The authors in [72] consider nonlinear quadratic system dynamics with input saturations and provide a method for synthesis of polynomial feedback control laws with maximized local stability region whose estimate is obtained from a generated quadratic Lyapunov function. Paper [2] considers switched systems and provides a method for synthesis of stabilizing switching controllers with H_∞ disturbance attenuation guarantee. The methodology which will be described in this chapter is capable of addressing nonlinear polynomial dynamics in constrained setting, with broad flexibility regarding the performance metrics which can be specified by the user.

The synthesis of a discrete-time controller for a continuous-time nonlinear system can be performed either in discrete-time by using a discretized plant model (e.g., obtained by forward Euler discretization) or in continuous time by using the continuous-time plant model and a subsequent approximate discrete-time controller implementation with a short sampling time. Although the method of this chapter can be developed in both conceptual frameworks, it will be presented here for the latter case in which the continuous-time nonlinear model is addressed without discretization. For this purpose, Section 2.2 describes a continuous-time version of the SOS stability certification from [43], which is an additional minor contribution of this chapter. The development dealing with discrete-time polynomial systems would involve the discrete-time version from [43]. Section 2.3 describes the two phases of the control synthesis method, and Section 2.4 demonstrates it on the bilinear model of a permanent magnet synchronous machine (PMSM) by synthesizing an EMPC controller for speed control of PMSM.

2.2 Continuous-time SOS Stability Verification

This section describes the continuous-time variation of the discrete-time SOS stability verification from [43]. The described stability certification will serve as a starting point for the development of the control synthesis technique in this chapter. Consider a continuous-time polynomial plant model:

$$\dot{x} = f_x(x,u), \tag{2.1a}$$
$$y = f_y(x), \tag{2.1b}$$

where $x \in \mathbb{R}^{n_x}$ is the state vector, $u \in \mathbb{R}^{n_u}$ the input vector, $y \in \mathbb{R}^{n_y}$ the output vector, $\dot{x} \in \mathbb{R}^{n_x}$ the derivative of the state, $f_x : \mathbb{R}^{n_x+n_u} \to \mathbb{R}^{n_x}$ the system function and $f_y : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ the output mapping. The functions f_x and f_y are assumed to be vector-valued multivariate polynomials, i.e., each component function of f_x and f_y is a multivariate polynomial in (x,u) and x, respectively. Consider as well an abstract form of the control law defined by the polynomial equalities and inequalities:

$$s = f_s(y;\eta), \tag{2.2a}$$
$$\mathbf{K}_{s,\eta} = \left\{\theta \mid \exists\lambda \text{ s.t. } h(s,\theta,\lambda;\eta) = 0,\ g(s,\theta,\lambda;\eta) \geq 0\right\}, \tag{2.2b}$$
$$u \in \kappa(\mathbf{K}_{s,\eta};\eta), \tag{2.2c}$$

where $s \in \mathbb{R}^{n_s}$ is the input to the controller, $\theta \in \mathbb{R}^{n_\theta}$ and $\lambda \in \mathbb{R}^{n_\lambda}$ internal variables, and the functions $f_s : \mathbb{R}^{n_y} \to \mathbb{R}^{n_s}$, $h : \mathbb{R}^{n_s+n_\theta+n_\lambda} \to \mathbb{R}^{n_h}$, $g : \mathbb{R}^{n_s+n_\theta+n_\lambda} \to \mathbb{R}^{n_g}$, and $\kappa : \mathbb{R}^{n_\theta} \to \mathbb{R}^{n_u}$ are vector-valued multivariate polynomials whose coefficients are parametrized in some way by the controller's tuning parameters $\eta \in \mathbb{R}^{n_\eta}$. While the goal of this chapter is to develop a method capable of finding a tuning η that optimizes some user-specified performance criteria, in this section the vector η is assumed fixed to some value and focus is on establishing SOS stability guarantees for the given closed-loop system.

Remark 1. *By the use of the KKT optimality conditions [11, 12, 15] whose equalities and inequalities can be embedded into the functions h and g in* (2.2b)*, a large variety of control structures can be written in the form of* (2.2)*, such as for example the EMPC control law as described in Section 2.4 or the PID with anti-windup as described in Section 3.4 of the next chapter.*

Remark 2. *The discrete-time formulation of SOS certification in reference [43] formulates the problem by including an additional subsystem, which for instance may correspond to an observer. In the formulation given here it is omitted in order to keep the expressions shorter.*

For selected tuning parameters η of the controller, the global closed-loop stability in the state variable x can be certified by a Lyapunov function V satisfying

$$-\|x\|_2^2 - \nabla_x V(x,\theta,\lambda)^T \dot{x} \geq 0, \tag{2.3a}$$
$$V(x,\theta,\lambda) - \|x\|_2^2 \geq 0. \tag{2.3b}$$

Considering the system dynamics (2.1) and the control law (2.2), the conditions (2.3) should be satisfied for all vectors

$$(x, \theta, \lambda, \dot{x}) \in \mathbf{T}_\eta \tag{2.4}$$

where the set

$$\mathbf{T}_\eta = \left\{(x, \theta, \lambda, \dot{x}) \mid \dot{x} = f_x(x, \kappa(\theta; \eta)),\ \hat{h}(x, \theta, \lambda; \eta) = 0,\ \hat{g}(x, \theta, \lambda; \eta) \geq 0\right\} \tag{2.5}$$

encapsulates the closed-loop system dynamics with the specified control law, and the functions

$$\hat{h}(x, \theta, \lambda; \eta) = h(f_s(f_y(x); \eta), \theta, \lambda; \eta), \tag{2.6a}$$

$$\hat{g}(x, \theta, \lambda; \eta) = g(f_s(f_y(x); \eta), \theta, \lambda; \eta), \tag{2.6b}$$

are introduced to make the notation lighter.

Remark 3. *Notice that in the equations* (2.3)-(2.5) *the $\dot{x}$ does not represent a derivative of the state x, but actually a variable denoted by $\dot{x}$ whose equality to $f_x(x, \kappa(\theta; \eta))$ is enforced by having the vector of variables $(x, \theta, \lambda, \dot{x})$ in the set $\mathbf{T}_\eta$.*

Remark 4. *The above formulation substitutes the system output y, the input to the controller s and the input to the system u by using equations* (2.1b) , (2.2a) *and* (2.2c)*, respectively. This formulation was chosen because it is characterized by a smaller number of equality constraints and allows a simpler exposition of the method. The treatment of the aforementioned relations as additional equality constraints in* (2.5)*, or elimination of some other variables by their substitution (e.g., elimination of the variable $\dot{x}$ by substituting it with* (2.1a)*) are straightforward alternatives.*

Remark 5. *The conditions* (2.3a) *and* (2.3b) *involve the terms $\|x\|_2^2$ whose weighting factors could be optimized as well. In this document, it will be proceed by keeping them fixed to unity for the sake of simplicity.*

The previous formulation involves the nonnegativity conditions (2.3) imposed over the set (2.5), which is a problem that can be tackled by using SOS programming. In particular, by denoting

$$\xi = (x, \theta, \lambda, \dot{x}) \tag{2.7}$$

and by restricting the Lyapunov function to be a polynomial of a certain (user-specified) degree, a sufficient condition for the nonnegativities (2.3) over the set $\mathbf{T}_\eta$ are the following

polynomial equalities

$$\begin{aligned} -\|x\|_2^2 - \nabla_x V(x,\theta,\lambda)^T f_x(x,u) &= \sigma_0(\xi) \\ &+ \sigma_1(\xi)^T \hat{g}(x,\theta,\lambda;\eta) + p_1(\xi)^T \hat{h}(x,\theta,\lambda;\eta) \\ &+ p_2(\xi)^T (\dot{x} - f_x(x,\kappa(x;\eta))), \end{aligned} \tag{2.8a}$$

$$\begin{aligned} V(x,\theta,\lambda) - \|x\|_2^2 &= \bar{\sigma}_0(\xi) \\ &+ \bar{\sigma}_1(\xi)^T \hat{g}(x,\theta,\lambda;\eta) + \bar{p}_1(\xi)^T \hat{h}(x,\theta,\lambda;\eta), \end{aligned} \tag{2.8b}$$

where the σ_0 and $\bar{\sigma}_0$ are SOS polynomials (defined below) with some user-specified degrees, $\sigma_1, \bar{\sigma}_1$ are vectors whose components are SOS polynomials with user specified-degrees, and p_1, p_2, $\bar{p}_1$ are vectors of arbitrary polynomials with user-specified degrees as well. A polynomial $\sigma(\xi)$ is said to be SOS if there exists a representation

$$\sigma(\xi) = v(\xi)^T P v(\xi), \tag{2.9}$$

where $v(\xi)$ is a vector of polynomials and $P \succeq 0$ is a positive semidefinite matrix of appropriate size, resulting in $\sigma(\xi)$ being nonnegative for every ξ. The satisfaction of the nonnegativity conditions in (2.3) over the set (2.5) follows directly from (2.8) since for $\xi \in \mathbf{T}_\eta$ the SOS polynomials σ are nonnegative and the arbitrary polynomials p are equal to zero.

The previous discussion of closed-loop stability verification thus boils down to the feasibility of the SOS problem

$$\begin{aligned} \text{find} \quad & V, \sigma_0, \sigma_1, p_1, p_2, \bar{\sigma}_0, \bar{\sigma}_1, \bar{p}_1 \\ \text{s.t.} \quad & \text{(2.8a), (2.8b)}, \\ & \sigma_0, \sigma_1, \bar{\sigma}_0, \bar{\sigma}_1 \quad \text{SOS polynomials}, \\ & V, p_1, p_2, \bar{p}_1 \quad \text{arbitrary polynomials}, \end{aligned} \tag{2.10}$$

where the decision variables are the coefficients of the polynomials $(V, \sigma_0, \sigma_1, p_1, p_2, \bar{\sigma}_0, \bar{\sigma}_1, \bar{p}_1)$. This problem converts to a semidefinite programming (SDP) convex optimization problem, and can thus be solved efficiently. The conversion can be done automatically by using freely available software like Yalmip [48]. For more information about the conversion of (2.10) to SDP, the reader is referred to, e.g., [45, 56].

Remark 6. *Although the method extends to the case with reference tracking, it is formulated here for the case of regulation of the state x to the origin of the state-space in order to avoid cumbersome expressions and better emphasize the fundamental concepts of the method. The application of the method to the case involving reference tracking in delta formulation is demonstrated in the computational example section.*

While the SOS program (2.10) ensures global stability in the state variable x, the method can

also be modified to address local stability over a set

$$\mathbf{X} = \{x \mid \psi_i(x) \geq 0, i = 1, \ldots, n_\psi\}. \tag{2.11}$$

where $\psi_i(x) : \mathbb{R}^{n_x} \to \mathbb{R}$ for all $i \in \{1, \ldots, n_\psi\}$ are polynomial functions. As described in [43], this is done by including the inequalities of the set $\mathbf{X}$ in the set $\mathbf{T}_\eta$, and by subsequently assigning them SOS polynomial multipliers (since they are inequality constraints) in (2.8) as well. The satisfaction of such a modified condition (2.8) does not guarantee invariance of the closed-loop system over the whole set $\mathbf{X}$, but only over the largest sublevel set of the Lyapunov function which is contained in $\mathbf{X}$.

2.3 Controller Synthesis

An attempt to involve the controller tuning parameters η as decision variables in the SOS programming stability verification (2.10) makes the problem loose the SDP structure, since after the conversion the constraint (2.8) would now be equivalent to a Bilinear Matrix Inequality (BMI) instead of to a Linear Matrix Inequality (LMI). The optimization problem thus ends up in a form which is not anymore an SDP.

The synthesis optimization problem addressed in what follows can be formulated as

$$\begin{aligned} &\text{min.} && \mathrm{P}(\eta) + \delta_{st}(\eta) \\ &\text{s.t.} && \eta \in \mathbf{D}, \end{aligned} \tag{2.12}$$

where the $\delta_{st}(\eta)$ is a function indicating the existence of the SOS stability certificate from (2.10):

$$\delta_{st}(\eta) = \begin{cases} 0, & \text{for } \eta \text{ with a stability certificate,} \\ +\infty, & \text{otherwise,} \end{cases} \tag{2.13}$$

the $\mathrm{P}(\eta)$ is a user-specified performance criteria for the closed-loop system with control parameters η, and the set $\mathbf{D}$ models some basic requirements on the tuning parameters η (e.g., a requirement that the Hessian in the cost function of a QP-based controller is symmetric positive definite)

The solving of optimization problem (2.12) in what follows consists of two phases. The first phase searches for feasible (i.e., stabilizing) control parameters η in (2.12). This is achieved by introducing a slack polynomial function into the stability certification constraint (2.8) (described in Section 2.3.1) and then by minimizing its presence by using Bayesian optimization to obtain parameters η feasible in (2.12) (described in Section 2.3.2). The second phase takes the generated stabilizing tuning parameters of (2.12) as initial conditions which are providing an indication about the location of a stabilizing region, and then starting from that data further explores for improvement of the performance criteria $\mathrm{P}(\eta)$ by means of Bayesian optimization and its data exploitation property (described in Section 2.3.3).

2.3.1 Reformulation with slack polynomial function

To allow controller synthesis, an additional SOS polynomial will be introduced in the condition (2.8a), denoted $\sigma_{\mathrm{sl}}(\xi)$ and of the same degree as the $\sigma_0(\xi)$, in order to play the role of a slack, which results in the constraint (2.8a) taking the form:

$$\begin{aligned} -\|x\|_2^2 - \nabla_x V(x,\theta,\lambda)^T f_x(x,u) &= \sigma_0(\xi) - \sigma_{\mathrm{sl}}(\xi) \\ &+ \sigma_1(\xi)^T \hat{g}(x,\theta,\lambda;\eta) + p_1(\xi)^T \hat{h}(x,\theta,\lambda;\eta) \\ &+ p_2(\xi)^T (\dot{x} - f_x(x,\kappa(x;\eta))), \end{aligned} \tag{2.14}$$

while the constraint (2.8b) will be retained without modification. Since any arbitrary polynomial can be written as a difference of two SOS polynomials [1], the $\sigma_0(\xi) - \sigma_{\mathrm{sl}}(\xi)$ term can express any arbitrary polynomial up to the degree of $\sigma_0(\xi)$ and $\sigma_{\mathrm{sl}}(\xi)$. Thus, the constraint consisting of (2.14) and (2.8b) has a feasible solution for any fixed value of the parameter η, provided that the degree of $\sigma_0(\xi)$ and $\sigma_{\mathrm{sl}}(\xi)$ is no smaller than the degrees of the other polynomials in (2.14).

Since the goal will be to minimize the presence of the slack as much as possible to make it become identically equal to zero, consider a cost function which is an integral of the SOS slack polynomial $\sigma_{\mathrm{sl}}(\xi)$. In particular, for $\sigma_{\mathrm{sl}}(\xi) = \sum_{i=1}^{n_\beta} \nu_i \beta_i(\xi)$ where ν_i are the polynomial's coefficients and $(\beta_i)_{i=1}^{n_\beta}$ the corresponding monomials, the integral over some simple set $\mathbf{Y}$ (like for instance a unit box which will be used in the numerical example) is:

$$\int_{\mathbf{Y}} \sigma_{\mathrm{sl}}(\xi) d\xi = \sum_{i=1}^{n_\beta} \nu_i \int_{\mathbf{Y}} \beta_i(\xi) d\xi, \tag{2.15}$$

and is a linear function in the coefficients ν_i weighted by the integrals of the β_i monomials over the set $\mathbf{Y}$ (these integrals are constant values).

Since this cost is a linear function in the coefficients of the SOS polynomial $\sigma_{\mathrm{sl}}(\xi)$, together with the constraint set containing (2.14) and (2.8b) it represents the SOS problem:

$$\begin{aligned} \mathrm{I}_\sigma(\eta) = \min. \quad & \int_{\mathbf{Y}} \sigma_{\mathrm{sl}}(\xi) d\xi & \\ \text{s.t.} \quad & (2.14), (2.8\mathrm{b}), & \\ & \sigma_{\mathrm{sl}}, \sigma_0, \sigma_1, \bar{\sigma}_0, \bar{\sigma}_1 & \text{SOS polynomials,} \\ & V, p_1, p_2, \bar{p}_1 & \text{arbitrary polynomials,} \end{aligned} \tag{2.16}$$

which corresponds to an SDP for any fixed η, and $\mathrm{I}_\sigma(\eta)$ is the optimal value of the problem. As $\sigma_{\mathrm{sl}}(\xi)$ is an SOS polynomial and thus globally nonnegative, the integral $\mathrm{I}_\sigma(\eta)$ is zero only when the polynomial $\sigma_{\mathrm{sl}}(\xi)$ is identically equal to zero. This $\mathrm{I}_\sigma(\eta) = 0$ corresponds to the case involving η for which the stability certificate from (2.10) exists. Otherwise, the slack function $\sigma_{\mathrm{sl}}(\xi)$ and its corresponding integral $\mathrm{I}_\sigma(\eta)$ are non-zero and minimization of $\mathrm{I}_\sigma(\eta)$ as a function of η would lead to stabilizing controller parameters (i.e., those satisfying $\mathrm{I}_\sigma(\eta) = 0$),

as discussed in the following section.

Instead of introducing an SOS polynomial $\sigma_{\mathrm{sl}}(\xi)$ in (2.14) to play the role of a slack, one can alternatively introduce an arbitrary (non-SOS) polynomial $p_{\mathrm{sl}}(\xi)$ whose presence in (2.16) can then be minimized by putting the ℓ_1 or ℓ_2-norm of the coefficients of $p_{\mathrm{sl}}(\xi)$ as a cost function. To demonstrate this flexibility, this chapter involves the SOS polynomial slack $\sigma_{\mathrm{sl}}(\xi)$ and the next chapter involves an arbitrary polynomial slack $p_{\mathrm{sl}}(\xi)$.

2.3.2 Search for stabilizing control parameters

Let $\mathbf{D}$ be the set of tuning parameters η satisfying some basic design requirements, as defined in (2.12). The set containing the tuning parameters η with SOS stability certificate is $\{\eta \mid \mathrm{I}_\sigma(\eta) = 0, \eta \in \mathbf{D}\}$. In the case when it is nonempty, it corresponds to the set of optimal solutions of the optimization problem

$$\begin{aligned} &\text{min.} && \mathrm{I}_\sigma(\eta) \\ &\text{s.t.} && \eta \in \mathbf{D}. \end{aligned} \tag{2.17}$$

In case of $\{\eta \mid \mathrm{I}_\sigma(\eta) = 0, \eta \in \mathbf{D}\}$ being empty, the optimal value of (2.17) would be larger than zero and it is not possible to find tuning parameters η with stability certificate (2.10).

The minimization (2.17) that leads to stabilizing tuning parameters $\{\eta \mid \mathrm{I}_\sigma(\eta) = 0, \eta \in \mathbf{D}\}$ can be performed by using a black-box global optimization method. In this work, the optimization problem (2.17) will be addressed by using Bayesian optimization [17], which is a derivative-free method for finding a constrained global optimal solution of a black-box cost function. The constraint set can be specified either explicitly (like the set $\mathbf{D}$ in (2.17)) or as an error in the evaluation of the cost (i.e., the value $+\infty$ returned by cost function), and the values of the cost function are allowed to be either deterministic or stochastic (see [49] for information pertaining to the practical aspects of the method). The algorithm is conceived in such a way that at each iteration of the Bayesian optimization method, the currently available cost evaluation pairs $\{\eta_i, \mathrm{I}_\sigma(\eta_\mathrm{i})\}$ are used to build a statistical model of the cost function based on Gaussian Processes [66]. This model is then employed to construct an acquisition function $a(\eta)$, which is such that its minimizer represents the next sampling point η that balances between exploitation of the currently known cost values and exploration of the less known regions of the cost function $\mathrm{I}_\sigma(\eta)$. The solving of (2.17) by Bayesian optimization can be accelerated by providing as initial conditions some tuning vectors η for which it can be believed that they are good candidates for being stabilizing. The feature that the next sampling point η is determined by minimizing the acquisition function $a(\eta)$ instead of operating with the actual cost function $\mathrm{I}_\sigma(\eta)$ makes the method particularly suitable for problems where the evaluation of the cost function $\mathrm{I}_\sigma(\eta)$ is time consuming or in some other sense expensive.

2.3.3 Optimization of performance

The optimization problem (2.12) can be addressed by Bayesian Optimization, which would treat the $+\infty$ values from $\delta_{st}(\eta)$ as the error in the evaluation of the cost. A problem however is that in the initial phase before any tuning parameters with SOS stability certificate are found (i.e., any parameter η with $\delta_{st}(\eta) = 0$), the Bayesian optimization would have only values $+\infty$ available, which are not very informative for choosing where to sample η next in order to reach a region with stabilizing parameters. For this reason, the solving of (2.12) by Bayesian optimization should be preceded by a search for stabilizing tuning parameters with (2.17), so that after a certain number (e.g., twenty) of stabilizing tuning parameters η is found by (2.17) one can use them as initial points to start Bayesian optimization on (2.12). These intial points (which would be different among themselves due to the exploration property) would provide some information to the Bayesian optimization solving (2.12) about the location of the stabilizing region in the space of tuning parameters, and by the exploitation property of Bayesian optimization it would be a region of focus for further investigation while minimizing the performance criteria $\mathrm{P}(\eta)$.

There is a great amount of flexibility in the choice of the cost term $\mathrm{P}(\eta)$ in (2.12), as it is allowed to be any performance criteria which can be evaluated for a fixed vector of tuning parameters η. A possible broadly applicable choice is an approximate evaluation of the integral of the infinite horizon trajectory cost over some set $\mathbf{W}$:

$$\mathrm{P}(\eta) = \int_{\mathbf{W}} C_{\infty,\eta}(x)\,dx, \quad C_{\infty,\eta}(x) = \int_0^{\infty} l(x(t), u(t))\,dt, \tag{2.18}$$

where $C_{\infty,\eta}(x)$ is the infinite horizon trajectory cost obtained with controller η when starting from the state x, and $l(x,u)$ is some stage cost. Equation (2.18) can be evaluated approximately by using Monte Carlo (MC) approximation for the integral and finite horizon approximations for the trajectory costs obtained from a discretized version of the continuous-time system:

$$\mathrm{P}(\eta) = \sum_{j=1}^{N_{mc}} \tilde{C}_{N_{st},\eta}(x_j), \quad \tilde{C}_{N_{st},\eta}(x) = T_s \sum_{k=0}^{N_{st}} l(x_k, u_k), \tag{2.19}$$

where T_s is the sampling time of the discrete-time simulation of the continuous-time system (obtained by applying the forward Euler method for example), x_k and u_k are the state and input values at the k-th simulation step, the $\tilde{C}_{N_{st},\eta}(x)$ is the finite horizon trajectory cost involving N_{st} simulation steps from the initial state x, and the N_{mc} is the number of samples from the set $\mathbf{W}$ in the MC approximation of the integral. As the evaluation of $\mathrm{P}(\eta)$ involves MC approximation, the Bayesian optimization should consider the cost function values as stochastic.

2.4 Computational Example

The control synthesis method will be demonstrated by an example involving a QP-based controller for control of a bilinear system with parametric uncertainty. The SOS programming problems are implemented by using Yalmip [48] as a modelling tool together with MOSEK as SDP solver, and the Bayesian optimization is applied by using Matlab's Statistics and Machine Learning Toolbox [49].

2.4.1 QP-based controller for a bilinear system with parametric uncertainty

This section synthesises a QP-based controller for speed control of a permanent magnet synchronous machine (PMSM). The small size of the QP control structure used in this section allows its implementation for the control of a bilinear PMSM model in EMPC fashion. In comparison to the MPC scheme for PMSM developed in [24], the controller synthesis of this section directly deals with the bilinear model of the system, thus circumventing the need for using a linear discrete-time prediction model like in [24] which is valid only at nominal (or some other fixed and in advance chosen) rotational speed. Furthermore, the controller presented here is synthesized robust to the stator resistance variations caused by temperature changes, and also without the outer speed-control loop based on an additional PI controller, which are features both mentioned in [24] as desirable to be addressed in future research.

The continuous-time model of the two-pole PMSM in the dq reference frame fixed to the rotor (see, e.g., [44]) has the form

$$\frac{dI_d(t)}{dt} = -\frac{\tilde{R}_s}{L_s} I_d(t) + \Omega_r(t) I_q(t) + \frac{1}{L_s} U_d(t), \tag{2.20a}$$

$$\frac{dI_q(t)}{dt} = -\frac{\tilde{R}_s}{L_s} I_q(t) - \left(I_d(t) + \frac{\Phi_0}{L_s} \right) \Omega_r(t) + \frac{1}{L_s} U_q(t), \tag{2.20b}$$

$$\frac{d\Omega_r(t)}{dt} = \frac{K_t}{J} I_q(t) - \frac{1}{J} \Gamma_{_L}(t), \tag{2.20c}$$

where $I_d(t)$ and $I_q(t)$ are the d and q component of the stator current vector $I(t) = [I_d(t), I_q(t)]^T$, $\Omega_r(t)$ is the rotational speed of the rotor, $U_d(t)$ and $U_q(t)$ are the d and q component of the input voltage vector $U(t) = [U_d(t), U_q(t)]^T$, $\Gamma_{_L}(t)$ is the load torque, the parameter $\tilde{R}_s$ is stator resistance, L_s stator inductance, Φ_0 the flux from the rotor's permanent magnet, K_t torque coefficient and J the rotational inertia of the rotor. The value of stator resistance $\tilde{R}_s$ is characterised by slow variations caused by the changes in temperature, leading to the values of $\tilde{R}_s$ which during operation can be several times larger than the R_s contained in Table 2.1 where the parameters of PMSM from [24] are given. The system has an input constraint concerning the magnitude of the input voltage vector:

$$\|U(t)\|_2 \leq U_{nom}, \tag{2.21}$$

while due to the thermal inertia of the machine, the stator current vector $I(t)$ is allowed to

Table 2.1 – PMSM parameters and nominal values, base values for the per-unit system, and the per-unit PMSM parameters.

Name	*Notation*	*Value*
Stator resistance	R_s	4.3 Ω
Stator inductance	L_s	3.56 mH
Flux from rotor	Φ_0	0.0245 Wb
Torque coefficient	K_t	36.8 mNm/A
Rotational inertia	J	$11 \cdot 10^{-7}$ Nm
Nominal (phase) voltage	U_{nom}	$36/\sqrt{3}$ V
Nominal current	I_{nom}	0.8 A
Nominal torque	Γ_{nom}	30 mNm
Base value of voltage	$U_b = U_{\text{nom}}$	$36/\sqrt{3}$ V
Base value of current	$I_b = I_{\text{nom}}$	0.8 A
Base value of load torque	$\Gamma_b = \Gamma_{\text{nom}}$	30 mNm
Base value of impedance	$Z_b = U_{\text{nom}}/I_{\text{nom}}$	25.98 Ω
Base value of speed	ω_b	$5000 \cdot 2\pi$ rad/s
Base value of rotor's flux	$\Phi_b = U_{\text{nom}}/\omega_b$	0.0397 Wb
Per-unit stator resistance	$r_s = R_s/Z_b$	0.4138
Per-unit stator inductance	$l_s = \omega_b L_s/Z_b$	0.0717
Per-unit flux from rotor	$\phi_0 = \Phi_0/\Phi_b$	0.6172
Per-unit torque constant	$\mathcal{K}_t = (K_t I_b)/(J\omega_b^2)$	51.11

make temporary violations of the constraint $\|I(t)\|_2 \le I_{nom}$ during transients, but still not of an excessively large magnitude which could cause damage on the machine or voltage source (e.g., it should be ensured that $\|I(t)\|_2 \le 5I_{nom}$).

By using the base values in Table 2.1, the following normalized model is obtained:

$$\frac{di_d(\tau)}{d\tau} = -\frac{\tilde{r}_s}{l_s} i_d(\tau) + \omega_r(\tau)\, i_q(\tau) + \frac{1}{l_s} u_d(\tau), \tag{2.22a}$$

$$\frac{di_q(\tau)}{d\tau} = -\frac{\tilde{r}_s}{l_s} i_q(\tau) - \left(i_d(\tau) + \frac{\phi_0}{l_s}\right)\omega_r(\tau) + \frac{1}{l_s} u_q(\tau), \tag{2.22b}$$

$$\frac{d\omega_r(\tau)}{d\tau} = \mathcal{K}_t i_q(\tau) - \frac{T_b}{J\omega_b^2}\gamma_{_L}(\tau), \tag{2.22c}$$

where $\tau = \omega_b t$ is per-unit time, $i_d(\tau) = I_d(t)/I_b$, $i_q(\tau) = I_q(t)/I_b$ are per-unit stator current components, $\omega_r(\tau) = \Omega_r(t)/\omega_b$ per-unit rotational speed, $u_d(\tau) = U_d(t)/U_b$, $u_q(\tau) = U_q(t)/U_b$ per-unit input voltage components, $\gamma_{_L}(\tau) = \Gamma_{_L}(t)/\Gamma_b$ per-unit load torque, and the per-unit parameters appearing in the model are as defined in Table 2.1. The input constraint for the per-unit model takes the form $\|u(\tau)\|_2 \le 1$, and the constraint on the current which can be temporarily violated during transients $\|i(\tau)\|_2 \le 1$. The corresponding state vector is $x(\tau) = [i_d(\tau), i_q(\tau), \omega_r(\tau)]^T$ and the input vector $u(\tau) = [u_d(\tau), u_q(\tau)]^T$. It can be seen that the

plant model is bilinear as it involves products of the state variables.

For the purpose of tracking a constant speed reference r, the steady-state target operating point $x_s = [i_{ds}, i_{qs}, \omega_{rs}]^T$, $u_s = [u_{ds}, u_{qs}]^T$ at which the rotational speed is equal to r is to be computed. In order to keep the synthesis example simpler and avoid additional complications, we consider the case involving a zero load torque (i.e., $\gamma_L = 0$) which results in the steady-state target operating point of the form

$$x_s = \begin{bmatrix} 0 \\ 0 \\ r \end{bmatrix}, \quad u_s = \begin{bmatrix} 0 \\ \phi_0 r \end{bmatrix}. \tag{2.23}$$

Consideration of a non-zero load torque, such as for example a γ_L which is a polynomial function of the rotational speed or a constant γ_L whose value is provided to the controller by an estimator, is also possible and reflects itself on the expression for the steady-state target (2.23).

The QP-based controller will be synthesised so that the QP takes as its inputs the target u_s and the deviation from the steady-state target $\Delta x = x - x_s$, and provides as its output (its optimal solution) the deviation Δu from the steady-state target u_s (i.e., the input signal to the PMSM is $u = u_s + \Delta u$). The form of the QP is selected to be

$$\begin{aligned} &\text{min.} && \tfrac{1}{2} z^T H z + \Delta x^T F z \\ &\text{s.t.} && Gz \leq d - Gu_s, \end{aligned} \tag{2.24}$$

where $z \in \mathbb{R}^2$ is the decision vector, $H \in \mathbb{R}^{2\times 2}$ is a symmetric positive definite matrix, $F \in \mathbb{R}^{3\times 2}$, and since the optimal solution of (2.24) corresponds to the input deviation Δu, the constraint matrices $G \in \mathbb{R}^{6\times 2}$ and $g \in \mathbb{R}^6$ are chosen such that they approximate the input constraint $\| u_s + \Delta u \|_2 \leq 1$ by an inner polytopic approximation consisting of $n_h = 6$ halfspaces $g_i^T z \leq d_i$, $\forall i \in \{1, \ldots, n_h\}$ where $g_i = [\cos(\pi i / n_h), \sin(\pi i / n_h)]^T$, $d_i = \cos(\pi / n_h)$, as can be seen in Fig. 2.2. The controller's tuning parameters (the vector η) are the elements of the H and F matrix, thus resulting in $\eta \in \mathbb{R}^9$ (due to the symmetry of H).

To represent the solution of the QP (2.24) as a system of polynomial equalities and inequalities, consider its corresponding KKT system [15]:

$$Hz + F^T \Delta x + G^T \lambda = 0, \tag{2.25a}$$
$$\lambda^T (Gz - d + Gu_s) = 0, \tag{2.25b}$$
$$d - Gz - Gu_s \geq 0, \tag{2.25c}$$
$$\lambda \geq 0, \tag{2.25d}$$

where $\lambda \in \mathbb{R}^6$ is a dual variable, (2.25a) represents the stationarity condition, (2.25b) is complementarity slackness, (2.25c) is primal feasibility and (2.25d) dual feasibility. The existence of λ so that (2.25) is satisfied by some z is a necessary and sufficient condition for that z to be

optimal in (2.24), provided some constraint qualification conditions are satisfied, which is the case for (2.24) with the specified G and d (for a detailed treatment of optimality conditions in convex optimization, see e.g. [12]).

For the controller synthesis, the vector of variables is selected to be

$$\xi = (r, \Delta x, \lambda, \Delta \dot{x}, \tilde{r}_s) \tag{2.26}$$

and all other variables are expressed as a function of ξ. In particular, at all places at which they appear, the x_s and u_s are expressed as in (2.23), the x and u as $x = x_s + \Delta x$ and $u = u_s + \Delta u$, respectively, the Δu as $\Delta u = z$, and the z as $z = -H^{-1}(F^T \Delta x + G^T \lambda)$ which is obtained from (2.25a).

For the purpose of reference tracking, the Lyapunov conditions (2.3) are formulated in delta space and have the form

$$-\Delta x^T \Delta x - \nabla_{\Delta x} V(\Delta x, r, \tilde{r}_s)^T \Delta \dot{x} \geq 0, \tag{2.27a}$$

$$V(\Delta x, r, \tilde{r}_s) - \Delta x^T \Delta x \geq 0. \tag{2.27b}$$

It can be seen that the Lyapunov function $V(\Delta x, r, \tilde{r}_s)$ depends on the speed reference r (specified below to be in the range $[-1, 1]$) and on the stator resistance $\tilde{r}_s$ (specified below to be in the range $[r_s, 5r_s]$, where r_s is as stated in Table 2.1). For control law (2.2), by selecting the input parameter $s = [\Delta x, u_s]^T$ and the internal variable $\theta = z$, the $h(s, z, \lambda)$ and $g(s, z, \lambda)$ polynomials take the form

$$h(s, z, \lambda) = \left[\lambda^T (G u_s + G z - d) \right], \tag{2.28a}$$

$$g(s, z, \lambda) = \begin{bmatrix} d - G(u_s + z) \\ \lambda \end{bmatrix}. \tag{2.28b}$$

The local stability certification set (2.11) is used to incorporate the bound on the reference $|r| \leq r_{\max}$ with $r_{\max} = 1$, the bound on the state vector $-x_{\max} \leq x_s + \Delta x \leq x_{\max}$ with $x_{\max} = [5, 5, 2]^T$, and the bound on the stator resistance $r_{s,\min} \leq \tilde{r}_s \leq r_{s,\max}$ with $r_{s,\min} = r_s$ and $r_{s,\max} = 5r_s$ where r_s is the parameter from Table 2.1. The set (2.11) is thus defined by

$$\psi = \begin{bmatrix} r + r_{\max} \\ -r + r_{\max} \\ x_s + \Delta x + x_{\max} \\ -x_s - \Delta x + x_{\max} \\ \tilde{r}_s - r_{s,\min} \\ -\tilde{r}_s + r_{s,\max} \end{bmatrix}. \tag{2.29}$$

The control synthesis is run with the Lyapunov function $V(\Delta x, r, \tilde{r}_s)$ of order 4, SOS σ and arbitrary p polynomial multipliers of order 2, and the SOS slack polynomial σ_{sl} of order 4. The

search ranges for tuning parameters in η are chosen to be $[-1,1]$ for each component, which is together with the positive-definiteness constraint for the matrix H embedded into the set $\mathbf{D}$. After 120 Bayesian Optimization iterations applied to (2.17) for the search of stabilizing solutions (involving a unit box as the set $\mathbf{Y}$ in (2.15)), 17 stabilizing tuning parameters η (i.e., vectors in the set $\{\eta \mid \mathrm{I}_\sigma(\eta) = 0, \eta \in \mathbf{D}\}$) were obtained, with an average time per Bayesian optimization iteration of about 9.5 minutes (the computing platform involved 3.0GHz Intel Core i7 processor and 16GB of RAM). These stabilizing parameters were then used as initial points in a total of 240 Bayesian optimization iterations applied to the performance optimization problem (2.12), with the average time per iteration slightly larger than 3 minutes. The performance criteria used was the MC approximation of the integral of trajectory costs (2.19). For it, the trajectories were simulated using forward Euler discretization with $T_s = 100\mu s$ (equivalent in per-unit to $\tau_s = \omega_b T_s$) with the value of $\tilde{r}_s$ fixed to $\tilde{r}_s = 2.5 r_s$. The N_{st} was selected to $N_{st} = 500$, and N_{mc} to $N_{mc} = 90$ which was spread equally for the reference values $r = 0$, $r = 0.8$ and $r = -0.8$. The stage cost was selected to be

$$l(x_k, u_k) = \Delta x_k^T Q_{sc} \Delta x_k + q_{sc} \| u_{k-1} - u_k \|_2^2, \tag{2.30}$$

where $Q_{sc} \in \mathbb{R}^{3\times 3}$ is diagonal with 2, 0.5 and 1 on its diagonal, and $q_{sc} = 1$. The set $\mathbf{W}$ used for initial states in (2.19) was selected identical to the locality constraint (2.29).

A slice of the control law with the obtained H and F matrix is represented in Fig. 2.1 with ω and r fixed to $\omega = 0$, $r = 1$. The controller was tested in simulation for various values of $\tilde{r}_s \in [r_s, 5r_s]$ by applying it with the sampling time $T_s = 100\mu s$ and starting it from many random initial points. Fig. 2.2 shows the state trajectory, the Lyapunov function values and the input signals obtained with speed reference $r = 1$ and a randomly generated initial state.

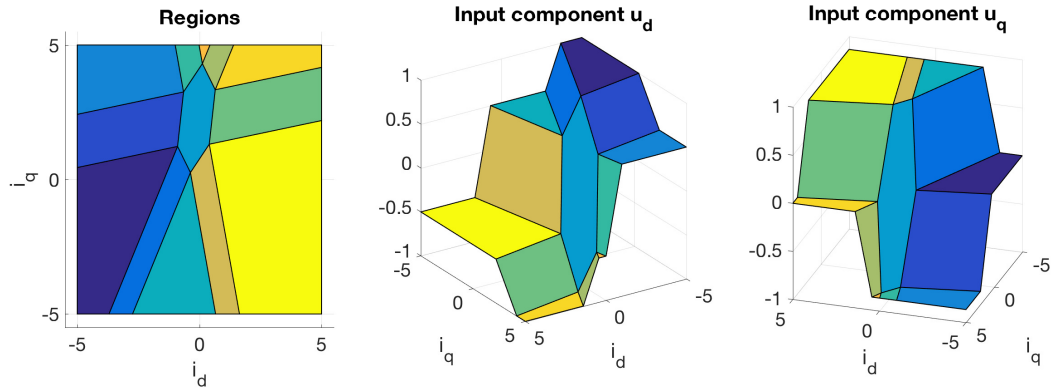


Figure 2.1 – The EMPC regions and the input components $u = [u_d, u_q]^T$ obtained with ω and r fixed to $\omega = 0$, $r = 1$. As can be seen, the number of regions of the EMPC is 13.

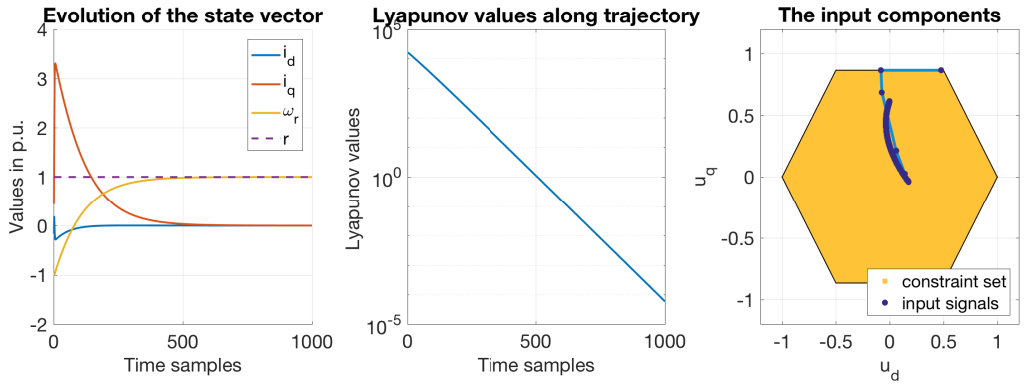


Figure 2.2 – Evolution of the state vector x starting from the initial $x_0 = [-0.166, 0.441, -0.998]^T$ for the reference $r = 1$ and resistance $\tilde{r}_s = r_s$, the corresponding Lyapunov values along the trajectory, and the input vectors $u = [u_d, u_q]^T$ along the trajectory (the target input is $u_s = [0, 0.617]^T$).

2.5 Conclusions

This chapter described an automatic tuning procedure capable of optimizing user-specified performance metric while imposing closed-loop stability guarantees. The functions describing the system are assumed to be of polynomial form and the closed-loop stability guarantees are established by using SOS stability verification for polynomial discrete-time systems, described in continuous-time in this chapter. The tuning method involves two phases, where the first phase searches for stabilizing controllers by minimizing a polynomial slack function introduced to the Lyapunov decrease condition and the second phase optimizes some user-specified performance criteria. The two phases are formulated as optimization problems which are tackled by applying Bayesian optimization as a black-box optimization technique. As an example of the possibilities that the tuning method provides, the presented synthesis method allows a design of EMPC controllers with closed-loop stability guarantees without relying on a terminal cost and/or constraint, and also without using the prediction horizon concept to formulate the control optimization problem. In particular, for a specified QP structure the tuning method can directly search for the stabilizing coefficients in the cost and/or the constraint set. The synthesis has been demonstrated on a numerical example involving an EMPC controller synthesized for speed control of a nonlinear PMSM model where the controller is synthesized robust to parametric uncertainty coming from the temperature-dependent stator resistance of the PMSM.

3 Extension to multimodel uncertainty and experimental verification

3.1 Introduction

This chapter brings several further developments related the automatic tuning method developed in Chapter 2. Besides demonstrating a broader applicability of the tuning method from Chapter 2 by applying it in a case involving a non-optimization-based, nonlinear control policy, the primary purpose of this chapter is to:

1. Experimentally demonstrate the validity of the method by application to a physical system.

2. Extend the method's practical computational capability to the case of multimodel plant uncertainty.

The mentioned broader generality of the method from Chapter 2 (i.e., applicability to non-optimization-based control schemes) can be anticipated from some of the illustrative examples provided in papers [43] and [42], from which the SOS stability certification used in Chapter 2 and this Chapter 3 originates. The anti-windup equipped PID control scheme considered in this chapter is embedded into the SOS framework by using the KKT conditions in a manner similar to an example present in [43]. This chapter, as a subsidiary contribution additional to the above two mentioned, also demonstrates an alternative possibility with respect to the method from Chapter 2 which consists of using a slack polynomial without imposed SOS form in the Lyapunov decrease condition, as described in Section 3.3.

The following Section 3.2 introduces notation and formulates the problem addressed by the chapter. Section 3.3 introduces the possibility of parallel multimodel uncertainty processing into the synthesis method of Chapter 2. The method is experimentally verified in Section 3.4 by its application to a PID with anti-windup whose parameters are tuned to be robustly stable with respect to the multimodel uncertainty in the considered mechanical experimental setup.

3.2 Problem Formulation

For the purpose of avoiding cumbersome notation and better emphasizing the fundamental aspects, the description will be done for the case involving regulation of the system state to the origin of the state-space. The applicability of the method in case of reference tracking can be achieved by shifting the origin of the state space and is demonstrated in application Section 3.4. The development in this chapter will be done in discrete-time, but is applicable in continuous-time framework as well.

Let $S_f = \{1, \dots, M\}$ denote the set containting indices of M different plant models. The actual physical plant is assumed to be a stationary system that corresponds to exactly one of the models $m \in S_f$, and its (unknown) index will be denoted by $\hat{m}$. Each of the models $m \in S_f$ is a discrete-time polynomial system:

$$x_m^+ = f_{mx}(x_m, u_m), \tag{3.1a}$$
$$y_m = f_{my}(x_m), \tag{3.1b}$$

where $x_m \in \mathbb{R}^{n_{mx}}$, $u_m \in \mathbb{R}^{n_u}$ and $x_m^+ \in \mathbb{R}^{n_{mx}}$ represent respectively the state vector, input vector and successor state of the model m, and $f_{mx} : \mathbb{R}^{n_{mx}+n_u} \to \mathbb{R}^{n_{mx}}$, $f_{my} : \mathbb{R}^{n_{mx}} \to \mathbb{R}^{n_y}$ are respectively the transition and output mapping of the model m. Notice that the inputs u_m and outputs y_m have dimensions which are independent of the model index $m \in S_f$ (the dimensions n_u and n_y, respectively), while the dimensions of the state vectors x_m are allowed to be different for various models $m \in S_f$. The mappings f_{mx} and f_{my} are assumed to be vector-valued multivariate polynomials $\forall\, m \in S_f$, i.e., each component function of f_{mx} and f_{my} is a multivariate polynomial in (x_m, u_m) and x_m, respectively.

During operation, the controller receives the output $y_{\hat{m}}$ of the actual physical plant $\hat{m}$ and computes the input $u_{\hat{m}}$ to be applied. As in Chapter 2, the control law will be described in an abstract form defined by the polynomial equalities and inequalities:

$$s = f_s(y_{\hat{m}}; \eta), \tag{3.2a}$$
$$\mathbf{K}_{s,\eta} = \left\{\theta \mid \exists \lambda \text{ s.t. } h(s, \theta, \lambda; \eta) = 0,\ g(s, \theta, \lambda; \eta) \geq 0\right\}, \tag{3.2b}$$
$$u_{\hat{m}} \in \kappa(\mathbf{K}_{s,\eta}; \eta), \tag{3.2c}$$

where $s \in \mathbb{R}^{n_s}$ is the input to the controller, $\theta \in \mathbb{R}^{n_\theta}$ and $\lambda \in \mathbb{R}^{n_\lambda}$ internal variables, and the functions $f_s : \mathbb{R}^{n_y} \to \mathbb{R}^{n_s}$, $h : \mathbb{R}^{n_s+n_\theta+n_\lambda} \to \mathbb{R}^{n_h}$, $g : \mathbb{R}^{n_s+n_\theta+n_\lambda} \to \mathbb{R}^{n_g}$, and $\kappa : \mathbb{R}^{n_\theta} \to \mathbb{R}^{n_u}$ are vector-valued multivariate polynomials whose coefficients are parametrized by the controller's tuning parameters $\eta \in \mathbb{R}^{n_\eta}$.

Remark 7. *As indicated in Chapter 2, by the use of the KKT optimality conditions [11, 15], a large variety of control structures can be written in the form of* (3.2)*, such as for example the EMPC control law as described Section 2.4, or the PID with anti-windup as demonstrated in the experimental Section 3.4 of this chapter.*

Since the index of the actual physical model $\hat{m}$ is unknown, the goal is to design the tuning η so that the controller (3.2) is closed-loop stabilizing $\forall m \in S_f$ and has some user-specified performance metric as good as possible. The closed-loop stability of the designed tuning vector η will be provided in this chapter by ensuring that for each of the models $m \in S_f$, there exists a polynomial Lyapunov function V_m such that

$$V_m(x_m, \theta_m, \lambda_m) - V_m(x_m^+, \theta_m^+, \lambda_m^+) - \|x_m\|_2^2 \geq 0, \tag{3.3a}$$

$$V_m(x_m, \theta_m, \lambda_m) - \|x_m\|_2^2 \geq 0, \tag{3.3b}$$

where θ_m and λ_m are the internal controller variables from (3.2) obtained when the actual system $\hat{m}$ corresponds to the system $m \in S_f$ involved in (3.3). In order to be able to enforce the conditions (3.3) locally over some set, they should hold for all vectors

$$\xi_m \in \Xi_{\mathbf{m}}, \tag{3.4}$$

where

$$\xi_m = (x_m, \theta_m, \lambda_m, x_m^+, \theta_m^+, \lambda_m^+) \tag{3.5}$$

is the vector containing all of the variables involved in (3.3), and the set

$$\Xi_{\mathbf{m}} = \{\xi_m \mid \psi_m(\xi_m) \geq 0\} \tag{3.6}$$

represents some user-specified local set of interest over which the Lyapunov inequalities (3.3) should hold for the model m, with the vector-valued functions $\psi_m(\xi_m)$, $m \in S_f$ assumed polynomial. A discussion of local stability for discrete-time systems can be found in Section 4.2.1 of Chapter 4.

3.3 Controller Synthesis with Parallel Multimodel Uncertainty Processing

The control synthesis method of this chapter involves two phases, as the original version of the method from Chapter 2. The first phase generates stabilizing tuning parameters and the second phase then searches for the tuning parameters that optimize some user-specified performance criteria.

3.3.1 Phase One - Search for stabilizing controller tunings

For the purpose of obtaining tuning parameters η characterized with the closed-loop stability guarantees of the form (3.3) for each of the models $m \in S_f$, slack polynomial functions $p_m(\xi_m)$, $m \in S_f$ are going to be introduced into the Lyapunov decrease conditions (3.3a) for each

$m \in S_f$:

$$V_m(x_m, \theta_m, \lambda_m) - V_m(x_m^+, \theta_m^+, \lambda_m^+) - \|x_m\|_2^2 \geq p_m(\xi_m), \tag{3.7a}$$

$$V_m(x_m, \theta_m, \lambda_m) - \|x_m\|_2^2 \geq 0, \tag{3.7b}$$

where the introduced slack $p_m(\xi_m)$ is of a degree no smaller than the polynomial on the left-hand side of (3.7a), and the relationship between the variables is defined by the control law (3.1) and plant dynamics (3.2). The introduced slack $p_m(\xi_m)$ can be imposed to be of SOS form whose integral over a unit box would be subsequently minimized, as in Chapter 2, but for the sake of demonstrating another possible variation of the synthesis method, the polynomial slack $p_m(\xi_m)$ will be taken in this chapter as an arbitrary polynomial whose ℓ_1-norm of the coefficients will be minimized, as described in what follows. Since the function on the left side of the inequality (3.7a) is polynomial and since the introduced slack $p_m(\xi_m)$ is a polynomial function of degree no smaller than the left side of the inequality, for any fixed tuning vector η there will always exist a polynomial $p_m(\xi_m)$ so that the inequalities (3.7) hold. The goal is thus to find the tuning parameters η for which the inequalities (3.7) hold with all of the polynomials $p_m(\xi_m)$ identically equal to zero (i.e., tuning η for which $p_m(\xi_m) \equiv 0$ for all $m \in S_f$).

By representing each of the slack polynomials in the form $p_m(\xi) = \sum_{i=1}^{n_{m\beta}} v_{mi}\beta_{mi}(\xi_m)$ where $(v_{mi})_{i=1}^{n_{m\beta}}$ are the polynomial's coefficients and $(\beta_{mi})_{i=1}^{n_{m\beta}}$ the corresponding monomials, consider the ℓ-1 norm of the $p_m(\xi_m)$ coefficients:

$$\|\operatorname{coef}\{p_m\}\|_1 = \sum_{i=1}^{n_{m\beta}} |v_{mi}|. \tag{3.8}$$

The search for stabilizing tuning parameters η can then be formulated as the following optimization problem:

$$\begin{aligned} \text{min.} \quad & \sum_{m \in S_f} \mathrm{I}_m(\eta) \\ \text{s.t.} \quad & \eta \in \mathbf{D}, \end{aligned} \tag{3.9}$$

where the set $\mathbf{D}$ models some basic requirements on the tuning parameters η (e.g., in case of a QP controller, models that the Hessian in the cost function should be symmetric positive definite), and the scalars $\mathrm{I}_m(\eta)$ are the optimal values of the optimization subproblems

$$\begin{aligned} \mathrm{I}_m(\eta) := \text{min.} \quad & \|\operatorname{coef}\{p_m\}\|_1 \\ \text{s.t.} \quad & \text{(3.7a), (3.7b) over set (3.6)}, \end{aligned} \tag{3.10}$$

As such, the $\mathrm{I}_m(\eta)$ values are quantifying for each $m \in S_f$ what is the smallest size slack polynomial $p_m(\xi_m)$ that has to be introduced in order to have the inequalities (3.7) satisfied. The polynomial form requirements introduced so far were imposed in order to allow addressing of the optimization problems of the form (3.10) by SOS programming techniques (see, e.g., [56], [45]), which as mentioned in Chapter 2 are able to exploit the polynomial structure to

convert the optimization problems (3.10) into semidefinite programming (SDP) forms for which efficient solution methods exist. The conversion of the problems such as (3.10) into a form that can be addressed by SOS programming is described in Chapter 2 (in the way done in [43], [42] and [61]) and will not be detailed in this chapter.

For any fixed value of η, the cost terms $\mathrm{I}_m(\eta)$ in the optimization problem (3.9) can be evaluated in parallel since their corresponding optimization problems (3.10) are mutually decoupled. The solving of (3.9) that leads to stabilizing tuning parameters can be performed by using a black-box global optimization method which can be run on a central unit with function evaluations $\mathrm{I}_m(\eta)$ performed in parallel for each fixed η. As in Chapter 2, the black-box optimization technique which will be used in the application example of this chapter is Bayesian optimization.

Remark 8. *Instead of the ℓ-1 norm, one could as well utilize other norms in* (3.10) *such as ℓ-2 or ℓ-∞ since they would also be convertible into the SDP optimization problem structure. In the description given here, the ℓ-1 norm is placed since it will be used in the application example.*

3.3.2 Phase Two - Optimization of the performance metric

While the first phase of the method generates tuning parameters with closed-loop stability guarantees, the second phase searches for the tuning parameters that have some user-specified performance metric as good as possible. The second phase can be represented as an optimization problem of the form

$$\begin{aligned} &\text{min.} \quad \sum_{m\in S_f} \mathrm{P}_m(\eta) + \delta_m(\eta) \\ &\text{s.t.} \quad \eta \in \mathbf{D}, \end{aligned} \tag{3.11}$$

where $\delta_m(\eta)$ is a function that indicates the existence of the closed-loop Lyapunov function V_m for the system m (i.e., indicates that for a given η the obtained optimal value in (3.10) is zero and thus $p_m(\xi_m) \equiv 0$):

$$\delta_m(\eta) = \begin{cases} 0, & \text{for } \eta \text{ with a stability certificate,} \\ +\infty, & \text{otherwise,} \end{cases} \tag{3.12}$$

and the $\mathrm{P}_m(\eta)$ is a user-specified performance criteria for the closed-loop system involving model $m \in S_f$ and tuning vector η.

The second phase of the synthesis method takes the stabilizing tuning parameters of (3.9) generated by the first phase (e.g., twenty stabilizing η vectors) and uses them as initial conditions for application of Bayesian optimization to (3.11). These intial points (which would be different among themselves due to the exploration property) would provide some information to the Bayesian optimization solving (3.11) about the location of the stabilizing region in the space of tuning parameters, and by the exploitation property of Bayesian optimization it

Table 3.1 – Identified coefficients of the models $G_m(z)$, $m \in \{1,2,3\}$.

m	b_{4m}	b_{3m}	b_{2m}	b_{1m}	b_{0m}
1	0	0.0148	0.0160	−0.0098	−0.0111
2	0	0.0141	0.0143	−0.0102	−0.0104
3	0	0.0142	0.0182	−0.0036	−0.0077
m	a_{4m}	a_{3m}	a_{2m}	a_{1m}	a_{0m}
1	1	−1.0660	−0.7239	1.0665	−0.2670
2	1	−1.1378	−0.6832	1.1414	−0.3129
3	1	−0.7408	−0.8223	0.7436	−0.1599

would be a region of focus for further investigation while minimizing the performance criteria.

3.4 Experimental Verification

This section demonstrates the control synthesis method on an example involving tuning of a PID controller with anti-windup so that it is robust to multimodel uncertainty. The SOS programming problems are implemented by using YALMIP [48] as a modelling tool together with MOSEK as SDP solver, and the Bayesian optimization is applied by using the Matlab's Statistics and Machine Learning Toolbox [49].

3.4.1 PID with anti-windup robust to multimodel uncertainty

The plant consists of three transfer functions of the form

$$G_m(z) = \frac{b_{4m}z^4 + b_{3m}z^3 + b_{2m}z^2 + b_{1m}z + b_{0m}}{a_{4m}z^4 + a_{3m}z^3 + a_{2m}z^2 + a_{1m}z + a_{0m}}, \tag{3.13}$$

where b_{lm}, a_{lm}, $l \in \{1,2,3,4\}$ are model coefficients for $m \in S_f = \{1,2,3\}$. The models $G_m(z)$ are obtained by the output error identification method applied on experimental data measured with $T_s = 20$ ms from one of the degrees of freedom of Quanser 3DOF Gyroscope (shown in Fig. 4.4) to which feedback-linearization was applied. The three models correspond to three operating points which involve rotational speeds of the gyroscope's disk of 300, 400 and 500 RPM. The identified parameters of the models $m \in S_f$ are given in Table 3.1. The input constraint is

$$-u_{\max} \le u \le u_{\max}, \tag{3.14}$$

where the input bound is $u_{\max} = 5$.

Figure 3.1 – The Quanser 3DOF Gyroscope used in the experiments.

The state-space model of $G_m(z)$, $m \in S_f$ is denoted as

$$x_m^+ = A_m x_m + B_m u_m, \tag{3.15a}$$
$$y_m = C_m x_m, \tag{3.15b}$$

where $A_m \in \mathbb{R}^{4\times4}$, $B_m \in \mathbb{R}^{4\times1}$, and $C_m \in \mathbb{R}^{1\times4}$ are the corresponding state-space matrices.

The system is controlled by a PID controller with anti-windup. In particular, the input u is computed as

$$u = K_p e + K_i u_i + K_d u_d \tag{3.16}$$

where $e = r - y$ is the tracking error of the reference r, $u_d = y - y^-$ is the derivative term being the difference between the current output y and the previous output y^-, and the integral term u_i is computed by using the anti-windup scheme which first updates the previous integrator state u_i^- to $\tilde{u}_i = u_i^- + e$ and then based on the value $u_{i,\text{tst}} = K_p e + K_i \tilde{u}_i + K_d u_d$ computes the new u_i as

$$u_i = \begin{cases} \frac{1}{K_i}(u_{\max} - K_p e - K_d u_d), \text{ if } u_{i,\text{tst}} > u_{\max}, \\ \tilde{u}_i, \text{ if } u_{\max} \le u_{i,\text{tst}} \le u_{\max}, \\ \frac{1}{K_i}(-u_{\max} + K_p e + K_d u_d), \text{ if } u_{i,\text{tst}} < -u_{\max}, \end{cases} \tag{3.17}$$

which prevents the violation of the input constraint (3.14). The synthesis is to be done with the vector of tuning parameters $\eta = [K_p, K_i, K_d]^T$.

In order to apply the synthesis technique, the control law must be represented as a system of polynomial equalities and inequalities. For this, the anti-windup part (3.17) can be represented as a projection on the set U_i:

$$u_i = \underset{v \in U_i}{\arg\min} \left\{ \tfrac{1}{2} \| v - \tilde{u}_i \| \right\}, \tag{3.18a}$$

$$U_i = \left\{ v \,\middle|\, \left| K_p e + K_i v + K_d u_d \right| \leq u_{\max} \right\}, \tag{3.18b}$$

for which the corresponding KKT system [15] is

$$u_i - \tilde{u}_i + \lambda_{\text{hi}} K_i - \lambda_{\text{lo}} K_i = 0, \tag{3.19a}$$

$$\lambda_{\text{hi}} (K_p e + K_i u_i + K_d u_d - u_{\max}) = 0, \tag{3.19b}$$

$$\lambda_{\text{lo}} (-K_p e - K_i u_i - K_d u_d - u_{\max}) = 0, \tag{3.19c}$$

$$-K_p e - K_i u_i - K_d u_d + u_{\max} \geq 0, \tag{3.19d}$$

$$K_p e + K_i u_i + K_d u_d + u_{\max} \geq 0, \tag{3.19e}$$

$$\lambda_{\text{hi}} \geq 0, \tag{3.19f}$$

$$\lambda_{\text{lo}} \geq 0, \tag{3.19g}$$

where $\lambda = [\lambda_{\text{hi}}, \lambda_{\text{lo}}]^T \in \mathbb{R}^2$ are dual variables, (3.19a) is the stationarity condition, (3.19b)-(3.19c) complementarity slackness, (3.19d)-(3.19e) primal feasibility and (3.19f)-(3.19g) dual feasibility.

Since the modelling of the control law for the purpose of computing (3.10) is an identical procedure for each $m \in S_f$, the index m will be omitted in the following equations in order to make the notation lighter. Due to the reference tracking, the stability certification will be done in delta space. For this purpose, steady-state target of x, u and u_i which cause the steady-state output y to be equal to a constant reference r will be computed. For a model $m \in S_f$, these target values will be denoted with subscript s as $(x_s.u_s, u_{is})$, and they are equal to

$$\begin{bmatrix} x_s \\ u_s \end{bmatrix} = \begin{bmatrix} (A - \mathbb{I}_{3\times 3}) & B \\ C & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0_{3\times 1} \\ r \end{bmatrix}, \tag{3.20a}$$

$$u_{is} = u_s / K_i. \tag{3.20b}$$

The variables denoting the deviations from the steady-state targets will be preceded by Δ (e.g., $\Delta x = x - x_s$).

For each $m \in S_f$, the vector of variables is selected as

$$\xi = (r, \Delta x, \Delta u_i^-, \Delta y^-, \lambda, \Delta u_i, \lambda^+) \tag{3.21}$$

and all other variables of the model m are expressed as a function of ξ. In particular, at all places at which they appear, the variables Δx^+ and Δu will be substituted by $\Delta x^+ = A\Delta x + B\Delta u$ and $\Delta u = K_p e + K_i(u_{is} + \Delta u_i) + K_d u_d - u_s$, the variables Δy, e, $\tilde{u}_i$, u_d by $\Delta y = C\Delta x$, $e = -\Delta y$, $\tilde{u}_i = u_{is} + \Delta u_i^- + e$, $u_d = \Delta y - \Delta y^-$, and the variables Δy^+, e^+, $\tilde{u}_i^+$, u_d^+ by $\Delta y^+ = C\Delta x^+$, $e^+ = -\Delta y^+$, $\tilde{u}_i^+ = u_{is} + \Delta u_i + e^+$, $u_d^+ = \Delta y^+ - \Delta y$.

The Lyapunov function conditions (3.7) in delta space, written without the slack polynomial $p_m(\xi)$, take the form

$$V(r, \Delta x, \Delta u_i^-, \Delta y^-, \lambda) - V(r, \Delta x^+, \Delta u_i, \Delta y, \lambda^+) \geq e^2,$$
$$V(r, \Delta x, \Delta u_i^-, \Delta y^-, \lambda) \geq \Delta x^T \Delta x,$$

where it can be seen that the Lyapunov function depends on the reference r. For control law (3.2), by selecting the input parameter $s = e$ and the internal variable $\theta = \Delta u_i$, and by considering the previously given expression for $\Delta u = K_p e + K_i(u_{is} + \Delta u_i) + K_d u_d - u_s$ as the output mapping κ, the $h(e, \Delta u_i, \lambda; \eta)$ and $g(e, \Delta u_i, \lambda; \eta)$ polynomials take the form

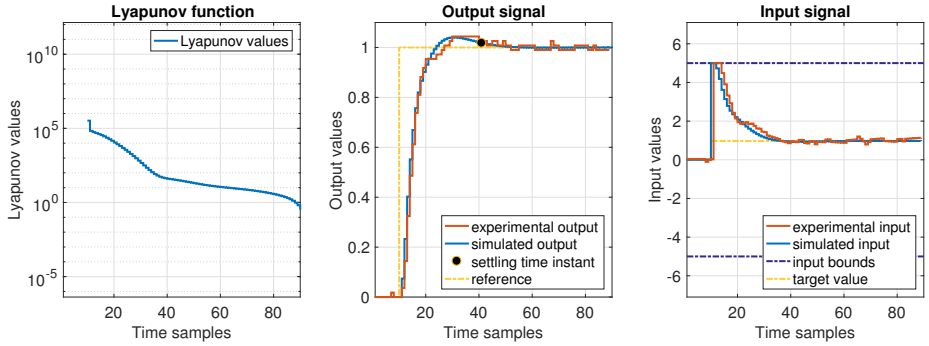
$$h = \begin{bmatrix} (u_{is} + \Delta u_i) - \tilde{u}_i + \lambda_{\text{hi}} K_i - \lambda_{\text{lo}} K_i \\ \lambda_{\text{hi}}(K_p e + K_i(u_{is} + \Delta u_i) + K_d u_d - u_{\max}) \\ \lambda_{\text{lo}}(-K_p e - K_i(u_{is} + \Delta u_i) - K_d u_d - u_{\max}) \end{bmatrix}, \tag{3.23a}$$

$$g = \begin{bmatrix} -K_p e - K_i u_i - K_d u_d + u_{\max} \\ K_p e + K_i u_i + K_d u_d + u_{\max} \\ \lambda_{\text{hi}} \\ \lambda_{\text{lo}} \end{bmatrix}. \tag{3.23b}$$

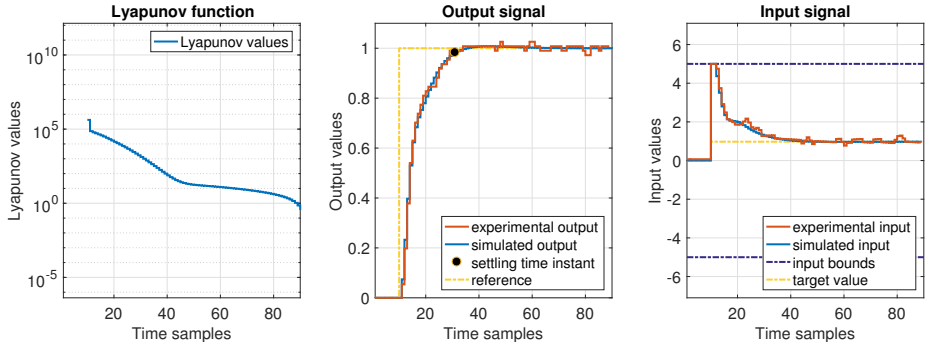
The locality set (3.6) is used to incorporate the bound on the reference $|r| \leq r_{\max}$ with $r_{\max} = 1.1$ and a bound on the derivative term for which the synthesis is performed by selecting $|\Delta y^- - \Delta y| \leq \Delta y_{\max}$ and $|\Delta y - \Delta y^+| \leq \Delta y_{\max}$. Note that $\Delta y_{\max}$ could also be an additional tuning parameter in η, but here it will be selected to be a constant value $\Delta y_{\max} = 20$. The set (3.6) is thus defined by

$$\psi = \begin{bmatrix} r + r_{\max} \\ -r + r_{\max} \\ \Delta y^- - \Delta y + \Delta y_{\max} \\ -\Delta y^- + \Delta y + \Delta y_{\max} \\ \Delta y - \Delta y^+ + \Delta y_{\max} \\ -\Delta y + \Delta y^+ + \Delta y_{\max} \end{bmatrix}. \tag{3.24}$$

The control synthesis is run with the Lyapunov function candidate $V(r, \Delta x, \Delta u_i^-, \Delta y^-, \lambda)$ of order four, SOS and arbitrary polynomial multipliers of order two and SOS slack polynomial of order four. The search ranges for parameters K_p, K_i and K_d are chosen to be $[0, 20]$, $[0, 5]$ and $[0, 2]$, respectively, which constitutes the set $\mathbf{D}$. After 60 Bayesian Optimization iterations applied for the search of stabilizing solutions (3.9), 36 stabilizing tuning parameters η (i.e.,



(a) Result obtained with minimization of squared tracking error, $\mathrm{P}_{\mathrm{SE}}(\eta)$.



(b) Result obtained with minimization of settling time, $\mathrm{P}_{\mathrm{ST}}(\eta)$.

Figure 3.2 – The Lyapunov function values (after the step change of reference at $k = 10$), the output signals and input signals (both simulated and experimentally measured results) of the obtained PID controllers. The displayed results correspond to model $m = 1$.

vectors in the set $\{\eta \mid \sum_{m\in S_f} \mathrm{I}_m(\eta) = 0, \eta \in \mathbf{D}\}$) were obtained with an average time per Bayesian optimization iteration (with sequential evaluation of $\mathrm{I}_m(\eta)$ in (3.9)) of about 7.4 minutes (the computing platform involved 3.0GHz Intel Core i7 processor and 16GB of RAM).

The obtained stabilizing tuning parameters were then used as initial points for Bayesian optimization applied to performance optimization problem (3.11). To investigate the influence of different performance metrics $\mathrm{P}(\eta)$ in (3.11), two cases were compared. The first case for the performance metric uses $\mathrm{P}_{m\mathrm{SE}}$ which is the sum of squared tracking errors of a step response, and the average over the values $\mathrm{P}_{m\mathrm{SE}}$, $m \in S_f$ will be denoted P_{SE}:

$$\mathrm{P}_{\mathrm{SE}}(\eta) = \frac{1}{3}\sum_{m=1}^{3} \mathrm{P}_{m\mathrm{SE}}(\eta), \quad \mathrm{P}_{m\mathrm{SE}}(\eta) = \sum_{k=0}^{\infty} e_m^2(k), \tag{3.25}$$

with $e_m(k)$ denoting the step response tracking error at time k for the model m. The second case for the performance metric uses the settling time (denoted by $\mathrm{P}_{m\mathrm{ST}}$) of the unit step response defined as

$$\mathrm{P}_{m\mathrm{ST}}(\eta) = \min\{k \mid \forall i \geq k, -0.01S_{\%} \leq e_{m,i} \leq 0.01S_{\%}\}$$

Table 3.2 – Comparison of obtained performances with two different choices for the performance metric in (3.11).

Minimization of the squared step tracking errors sum, $P_{mSE}(\eta)$		
	Settling time sample	Sum of squared errors
Model 1	32	3.258
Model 2	34	3.321
Model 3	30	3.575
Average	32	3.385

Minimization of settling time, $P_{mST}(\eta)$		
	Settling time sample	Sum of squared errors
Model 1	22	3.586
Model 2	22	3.656
Model 3	19	3.824
Average	21	3.689

where the settling criterion was selected to be $S_{\%} = 2\%$. The average settling time will be denoted by P_{ST}:

$$P_{ST}(\eta) = \frac{1}{3}\sum_{m=1}^{3} P_{mST}(\eta), \tag{3.26}$$

The computation of $P_{mSE}(\eta)$ and $P_{mST}(\eta)$ was done by simulations of the unit step response for $N_{st} = 1000$ steps. The performance optimization problem was run with 120 Bayesian Optimization iterations with $P_{mSE}(\eta)$ and 120 iterations with $P_{mST}(\eta)$. The average time per Bayesian iteration (sequential evaluation of P_{mSE} and P_{mST} in (3.11)) was about 1.7 minutes in the case of P_{mSE} and 1.4 minutes in the case of P_{mST}.

The simulation and experimental results for the step response in case of the model $m = 1$ are shown in Fig. 3.2. From the figure, it can be seen that the experimental measurement well corresponds to the simulated behaviour. Table 3.2 shows the performance of the two obtained controllers, where it can be seen that the $P_{SE}(\eta)$ as the performance metric involves an 8.24% smaller average value of the sum of the squared tracking errors (3.385 in comparison to 3.689), and $P_{ST}(\eta)$ causes a 34.4% smaller average settling time of the step response (21 samples in comparison to 32). To reduce the influence of noise, the results in the table correspond to an average of five consecutively performed step experiments.

3.5 Conclusions

This chapter describes an extension of the tuning method from Chapter 2 to cases involving multimodel plant uncertainty and provides an experimental validation of the method by its

application to an experimental mechanical system consisting of a feedback-linearized Quanser 3DOF Gyroscope. The chapter more prominently highlights the applicability of the synthesis method from Chapter 2 to non-optimization-based control structures by demonstrating it in case involving a PID with anti-windup which is in the considered experimental setup synthesized robust to multimodel uncertainty. The chapter as well demonstrates an alternative possibility for the method of Chapter 2 which consists of removing the SOS requirement on the introduced slack polynomial.

4 Automatic tuning based on scenario approach technique

4.1 Introduction

In order to extend the benefits of the automatic tuning procedure presented in Chapter 2 to problems of larger size, instead of using the SOS programming technique for computation of Lyapunov functions, a method for numerical Lyapunov function estimation based on scenario approach will be introduced in this chapter. This method allows a high accuracy numerical estimation of Lyapunov functions in problems of much larger size than those that can be addressed by the SOS programming. Nevertheless, the better scalability is achieved at a cost of being able to compute the Lyapunov functions only up to some finite accuracy, as a consequence of which the Lyapunov functions computed by the method of this chapter should be referred to as numerical estimates of Lyapunov functions, as described in more detail in what follows.

The method of this chapter is characterized by a flexibility that allows its application to a broad variety of nonlinear system dynamics, controller structures, and user-specified performance metrics. In addition to its ability to address problems of much larger size than the approach of Chapter 2, it also does not require a polynomial form for the functions describing the system. Instead of using SOS programming as the method of Chapter 2, the generation of closed-loop Lyapunov functions is formulated in Section 4.2 as a robust optimization problem that is then tackled by applying a Chebychev center [15] and scenario-based optimization [19, 20, 21] to reformulate it into a linear programming (LP) optimization form. This Lyapunov function generation technique is then used in Section 4.3 where the controller synthesis is formulated as an optimization problem that can be tackled by using black-box optimization techniques, such as Bayesian optimization which is used in the application examples of this chapter.

The performance of the method is demonstrated on two examples in Section 4.4 which cannot be addressed by the SOS-based approach due to scalability and modelling limitations. The first example involves the synthesis of an early-terminated optimization-based controller where an extrapolated gradient-projection optimization algorithm with approximate evaluation of projections is applied to a nonlinear control problem corresponding to a soft-constrained

speed control of a (bilinear) permanent magnet synchronous machine (PMSM) model. It is interesting to note that the early-terminated optimization algorithm employed in this example does not involve mathematically rigorous convergence guarantees when run for an infinite amount of time, but it does not represent a problem as the method's focus is directly on the obtained control policy and its consequent closed-loop behavior which are shaped by tuning as well the optimization algorithm parameters (the stepsize and extrapolation factor) together with the other available tuning parameters, as demonstrated in the example. The SOCP is formulated without using the prediction horizon concept (similarly as in the EMPC example of Section 2.4) and is actually obtained by assuming an SOCP structure whose coefficients are then tuned (together with the optimization algorithm's tuning parameters) by the method in order to shape the control policy and its consequent closed-loop behavior. The second example involves synthesis of a cascaded linear controller with saturations for constrained control of a nonlinear plant, where the nonlinear plant dynamics correspond to a gyroscope whose rotating disk's position is to be controlled. The controller of this example is synthesized directly for the nonlinear plant dynamics (without performing linearizations) of the mechanical gyroscopic system.

The method of this chapter does not necessarily require for its application an analytical expression of the plant dynamics as, for example, the method of Chapter 2 requires an analytical expression of polynomial form. Thanks to the utilization of the scenario approach, the method needs only plant model simulations/evaluations in order to perform the controller synthesis. Controller design methods that require only a simulator of the plant dynamics (i.e., do not require an analytical expression for the plant model) can also be found in the field of dynamic programming [7, 8]. These methods belong to the class of approximate dynamic programming, also known under the names neuro-dynamic programming [10] and reinforcement learning, and involve methods such as the rollout algorithm, approximation in value space (e.g., Q-learning algorithm, approximate policy iteration) and approximation in policy space. The approximation in policy space can be considered as the class closest to the method of this chapter, since it involves selection of a problem-related parametrized control policy whose parameters are then searched for by minimizing a performance cost function using the gradient method or some other algorithm such as random search. In the field of reinforcement learning, the methods of this kind are known as policy search [28] and are classified to the methods that (instead of a simulator) use trajectories experimentally obtained from the physical plant (class known as model-free methods) and the methods that use a combination of a simulator and experimentally generated trajectories which iteratively refine the simulator's accuracy by gradually providing more experimental data (class known as model-based methods) [62]. For the purpose of the maximization of a performance metric (which is called a reward function), the policy search reinforcement learning methods in some cases also utilize Bayesian optimization [17, 46, 74], which is the black-box optimization technique used in this chapter as well. While the aforementioned methods commonly use uncertain simulation-based plant models (which are in the case of model-based reinforcement learning iteratively refined as the algorithm progresses), the method of this chapter requires availability of an accurate

deterministic simulation-based model and places focus on obtaining a controller for which the Lyapunov function can be generated by using a Chebychev center and scenario-based optimization, as described in Section 4.2. An application of the scenario approach for stability verification was previously considered in case of robust linear systems control by the authors in [19].

In comparison to the tuning method of Chapter 2, the method of this chapter involves advantages concerning both the problem scalability and modelling flexibility. In particular, the Lyapunov function generation method to be presented in Section 4.2 allows one to circumvent the need for employing a sum-of-squares (SOS) stability certification technique [42, 43, 56]. This makes the tuning method applicable to larger problems (in terms of the controller and plant size), as demonstrated in the PMSM example of Section 4.4.1 whose addressing by SOS techniques would not be possible due to SOS scalability limitations. Moreover, the method of this chapter also involves better modelling flexibility since it does not require a polynomial form for the functions describing the system, as demonstrated in the example of Section 4.4.2 where the involved gyroscope dynamics contain products of states and trigonometric functions of states.

The rest of the chapter is structured as follows. Section 4.2 describes the Lyapunov function generation technique which is then used in the synthesis method described in Section 4.3. The method is then demonstrated on two examples provided in Section 4.4, and Section 4.5 contains conclusions.

4.2 Generation of Lyapunov functions based on Chebychev center and scenario-based approach

This section describes a technique for generation of high accuracy Lyapunov function numerical estimates that will be used by the controller synthesis method. For both the continuous-time and the discrete-time settings, the problem of generating a Lyapunov function is formulated as a robust optimization problem which is then reformulated into a linear programming (LP) form by applying a Chebychev center technique and scenario programming approach. To avoid cumbersome expressions and to better emphasise the fundamental aspects, the description of this section is done for the case involving regulation of the system state to the origin of the state-space, while the extension to the reference tracking based on shifting of the state-space origin is demonstrated in the application examples Section 4.4.

4.2.1 Problem formulation

Continuous-time setting

The plant model has the form

$$\dot{x} = f_x(x, u), \tag{4.1a}$$
$$y = f_y(x), \tag{4.1b}$$

where $x \in \mathbb{R}^{n_x}$ is the plant state vector, $u \in \mathbb{R}^{n_u}$ the input vector, $y \in \mathbb{R}^{n_y}$ the output vector, $\dot{x} \in \mathbb{R}^{n_x}$ the derivative of the state, and $f_x : \mathbb{R}^{n_x+n_u} \to \mathbb{R}^{n_x}$ and $f_y : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ are the system function and output mapping, respectively.

The control law can be modeled as a dynamical system

$$\dot{z} = f_z(z, y; \eta), \tag{4.2a}$$
$$u = f_u(z; \eta), \tag{4.2b}$$

where $z \in \mathbb{R}^{n_z}$ is the controller state vector, $\dot{z} \in \mathbb{R}^{n_x}$ the derivative of the state, $f_z : \mathbb{R}^{n_z+n_y} \to \mathbb{R}^{n_z}$ and $f_u : \mathbb{R}^{n_z} \to \mathbb{R}^{n_y}$ represent respectively the controller dynamics function and output mapping, and $\eta \in \mathbb{R}^{n_\eta}$ is a vector containing the controller's tuning parameters. Within this Section 4.2, the tuning parameters of the controller η are assumed fixed to some value, and the goal is to determine whether for that particular choice of η a Lyapunov function can be generated (the tuning of η will be the object of focus in Section 4.3). For easier notation in what follows, a vector $\theta \in \mathbb{R}^{n_\theta}$ with $n_\theta = n_x + n_z$ is defined by

$$\theta = (x, z), \tag{4.3}$$

which represents the state of the overall closed-loop system comprising the aforementioned plant (4.1) and controller (4.2). Its dynamics are thus defined by

$$\dot{\theta} = f_{cl}(\theta; \eta) \tag{4.4}$$

where

$$f_{cl}(\theta) = \begin{bmatrix} f_x\left(x, f_u(z; \eta)\right) \\ f_z\left(z, f_y(x); \eta\right) \end{bmatrix}. \tag{4.5}$$

The goal is to certify local stability of the origin $\theta = 0$ for the closed-loop system (4.4) by finding a continuously differentiable Lyapunov function $V(\theta)$ over a compact connected region of interest Ξ that includes $\theta = 0$ in its interior. Besides being zero at the origin:

$$V(0) = 0, \tag{4.6}$$

the Lyapunov function $V(\theta)$ should satisfy

$$-\nabla V(\theta)^T f_{cl}(\theta) \geq \|\theta\|_2^2, \quad \forall \theta \in \Xi, \tag{4.7a}$$

$$V(\theta) \geq \|\theta\|_2^2, \quad \forall \theta \in \Xi. \tag{4.7b}$$

To ensure solution existence and uniqueness of the system of differential equations (4.4), the system function $f_{cl}(\theta)$ is assumed locally Lipschitz continuous over a region $D \supset \Xi$, which means that for every point $\tilde{\theta} \in D$ there is a neighbourhood $\tilde{D}$ around $\tilde{\theta}$ such that there holds

$$\left\| f_{cl}(\tilde{\theta}_a) - f_{cl}(\tilde{\theta}_b) \right\| \leq \tilde{L} \left\| \tilde{\theta}_a - \tilde{\theta}_b \right\|, \quad \forall \tilde{\theta}_a, \tilde{\theta}_b \in \tilde{D} \tag{4.8}$$

where $\tilde{L} > 0$ is a positive constant that depends on $\tilde{\theta} \in D$.

Under the above conditions, the existence of a Lyapunov function $V(\theta)$ ensures that the equilibrium pont $\theta = 0$ (i.e., the point $\theta = 0$ satisfying $0 = f_{cl}(0;\eta)$) is locally asymptotically stable; that is, there is a neighbourhood of $\theta = 0$ such that all trajectories $\theta(t)$ started within it satisfy

$$\lim_{t \to \infty} \theta(t) = 0 \tag{4.9}$$

and do not leave the neighbourhood. The following theorem (adapted from [41], Theorem 3.3) summarizes the above statements for the asymptotic local stability of $\theta = 0$.

Theorem 1. *Let $\theta = 0$ be an equilibrium point for the closed-loop system (4.4) whose system function $f_{cl}(\theta)$ is locally Lipschitz continuous over a set D that contains a compact connected region $\Xi \subset \mathbb{R}^{n_\theta}$ whose interior includes $\theta = 0$. Let $V : \mathbb{R}^{n_\theta} \to \mathbb{R}$, $V(0) = 0$ be a continuously differentiable function that satisfies the Lyapunov conditions (4.7). Then, the equilibrium point $\theta = 0$ is locally asymptotically stable.*

Remark 9. *The formulation of this section assumes that the Lyapunov function $V(\theta)$ is a function of the state θ. It should be noted that the method to be presented also allows the Lyapunov function to be a function of other variables that can be available, such as for example the dual variables of the primal-dual optimal solution pair provided by the controller's optimization algorithm (assuming they are uniquely determined for each θ) or the slack variables of the optimization-based controller with soft constraints.*

Remark 10. *From the point of view of the method to be presented, the terms $\|\theta\|_2^2$ in (4.7) imposing the decrease and positivity of the Lyapunov function can be replaced by functions of more general form and can as well be weighted by positive weighting factors which can be searched for by the method. For the sake of simplicity of the exposition, the terms $\|\theta\|_2^2$ are used, which as well turns out to suffice for the application examples given in this chapter..*

Discrete-time setting

The plant model has the form

$$x^+ = f_x(x,u), \tag{4.10a}$$
$$y = f_y(x), \tag{4.10b}$$

where $x \in \mathbb{R}^{n_x}$ is the plant state vector, $u \in \mathbb{R}^{n_u}$ the input vector, $y \in \mathbb{R}^{n_y}$ the output vector, $x^+ \in \mathbb{R}^{n_x}$ the plant's successor state, and $f_x : \mathbb{R}^{n_x+n_u} \to \mathbb{R}^{n_x}$ and $f_y : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ are the plant transition and output mapping, respectively.

The control law can be modeled as a dynamical system

$$z^+ = f_z(z,y;\eta), \tag{4.11a}$$
$$u = f_u(z;\eta), \tag{4.11b}$$

where $z \in \mathbb{R}^{n_z}$ is the controller state vector, $z^+ \in \mathbb{R}^{n_x}$ the controller successor state, $f_z : \mathbb{R}^{n_z+n_y} \to \mathbb{R}^{n_z}$ and $f_u : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ represent respectively the controller transition and output mapping, and $\eta \in \mathbb{R}^{n_\eta}$ is a vector containing the controller's tuning parameters. As noted in the description of the continuous-time setting, within this Section 4.2 the tuning parameters η of the controller are assumed fixed to some value and the goal is to determine whether for that particular choice of η a Lyapunov function can be generated. The closed-loop system state vector $\theta \in \mathbb{R}^{n_\theta}$ with $n_\theta = n_x + n_z$ is defined by

$$\theta = (x,z), \tag{4.12}$$

which represents the state of the closed-loop interconnection of the plant (4.10) and controller (4.11). The closed-loop transition mapping is defined by

$$\theta^+ = f_{cl}(\theta;\eta) \tag{4.13}$$

where

$$f_{cl}(\theta) = \begin{bmatrix} f_x\left(x, f_u(z;\eta)\right) \\ f_z\left(z, f_y(x);\eta\right) \end{bmatrix}. \tag{4.14}$$

The goal is to certify local stability of the origin of the closed-loop system (4.13) by finding a continuous Lyapunov function $V(\theta)$ over a compact connected region of interest Ξ that includes the origin $\theta = 0$ in its interior. A function $V(\theta)$ should satisfy $V(0) = 0$ as well as

$$V(\theta) - V(\theta^+) \geq \|\theta\|_2^2, \quad \forall \theta \in \Xi, \tag{4.15a}$$
$$V(\theta) \geq \|\theta\|_2^2, \quad \forall \theta \in \Xi, \tag{4.15b}$$

The following theorem summarizes a set of sufficient conditions that provide local asymptotic stability of the equilibrium point $\theta = 0$ (i.e., of the point $\theta = 0$ satisfying $0 = f_{cl}(0;\eta)$), where

the local asymptotic stability means that there is a neighbourhood of $\theta = 0$ such that for every sequence $\{\theta_k\}$ started within it there holds

$$\lim_{k\to\infty} \theta_k = 0 \tag{4.16}$$

without any element of the sequence $\{\theta_k\}$ leaving the neighbourhood. The theorem involves an assumption on (4.13) that the difference between any two subsequent states θ and θ^+ is no larger than an upper bound $q > 0$:

$$\left\|\theta - \theta^+\right\|_2 \leq q, \quad \forall \theta \in \Xi, \tag{4.17}$$

and also that the set of all interior points of Ξ whose distance to the closest boundary point of Ξ is larger than q:

$$\Xi_q = \left\{\theta \mid \theta \in \mathrm{int}(\Xi), \inf_{\gamma\in\mathrm{bnd}(\Xi)} \left\|\theta - \gamma\right\|_2 > q\right\} \tag{4.18}$$

is nonempty, where int(Ξ) and bnd(Ξ) denote the set of all interior points and boundary points of Ξ, respectively. The condition (4.17) is convenient in cases of discrete-time dynamics obtained by discretization of continuous-time systems since in these cases faster sampling times result in reduced distances between pairs θ and θ^+.

Theorem 2. *Let $\theta = 0$ be an equilibrium point for the closed-loop system* (4.13) *whose system function $f_{cl}(\theta;\eta)$ satisfies the condition* (4.17) *over a compact connected region $\Xi \subset \mathbb{R}^{n_\theta}$ which has $\theta = 0$ in its interior. Assume that the set Ξ_q defined in* (4.18) *is nonempty. Let $V : \mathbb{R}^{n_\theta} \to \mathbb{R}$, $V(0) = 0$ be a continuous function that satisfies the Lyapunov conditions* (4.15)*. Then, the equilibrium point $\theta = 0$ is locally asymptotically stable.*

Proof. Consider a set difference of Ξ and Ξ_q denoted as Ξ_s:

$$\Xi_s = \Xi \setminus \Xi_q, \tag{4.19}$$

which can be geometrically envisioned as a shell of thickness q which encloses the set Ξ_q. Consider the infimum of the Lyapunov function $V(\theta)$ over the set Ξ_s:

$$a = \inf_{\tilde{\theta}\in\Xi_s} \left\{V(\tilde{\theta})\right\}. \tag{4.20}$$

Since the set Ξ_s is compact and $V(\theta)$ is continuous, by the Weierstrass theorem the infimum is attained at some point, and due to (4.15b) the value a is strictly positive. For some $b \in (0, a)$, consider the level set of the Lyapunov function

$$\Omega_b = \left\{\theta \mid V(\theta) \leq b, \theta \in \Xi\right\}. \tag{4.21}$$

Since $a > 0$ and $V(0) = 0$, due to continuity of V, the set Ω_b is nonempty. For an initial state $\theta_0 \in \Omega_b$, the subsequent states θ_k, $k = 1, 2, \ldots$ do not leave the bounded set Ξ_q. This can be

shown by contradiction, since if for some state $\theta_k \in \Xi_q$ there happens $\theta_{k+1} \notin \Xi_q$, it would imply $\theta_{k+1} \in \Xi_s$ due to (4.17). Since θ_k belongs to a sequence started with θ_0 for which $V(\theta_0) \leq b$, due to (4.15a) there holds $V(\theta_k) \leq b$, and since $\theta_{k+1} \in \Xi_s$ there must hold $V(\theta_{k+1}) \geq a$ due to (4.20), resulting in $V(\theta_k) \leq b < a \leq V(\theta_{k+1})$ contradicting the condition (4.15a).

To show asymptotic stability of sequence $\{\theta_k\}$ started with $\theta_0 \in \Omega_b$ in addition to the sequence's boundedness, there should hold $\lim_{k\to\infty} \theta_k = 0$, which means that for every ball of radius $\epsilon > 0$ around 0, denoted as $\mathscr{B}_\epsilon(0) = \{\theta \mid \|\theta\|_2 \leq \epsilon\}$, there should be an integer $K > 0$ such that $\forall\, k \geq K$, $x_k \in \mathscr{B}_\epsilon(0)$. This can be shown by contradiction. For a given ϵ, let

$$\gamma = \inf_{\tilde{\theta} \in \Xi_q \setminus \mathscr{B}_\epsilon(0)} \left\{ \|\tilde{\theta}\|_2^2 \right\}, \tag{4.22}$$

which is the smallest Lyapunov decrease guaranteed by (4.15a) over the set $\Xi_q \setminus \mathscr{B}_\epsilon(0)$. Let sequence $\{\theta_k\}$ be started with $\theta_0 \in \Omega_0$. To arrive at a contradiction, assume it has an infinite subsequence $\{\theta_m\}_\kappa$, $m = 0, 1, \ldots$ such that $\theta_m \in \Xi_q \setminus \mathscr{B}_\epsilon(0)$ for all m. For each θ_m, there holds $V(\theta_m) - V(\theta_{m+1}) \geq \gamma$, which is an inequality that follows from (4.22) because $\{\theta_m\} \subset \Xi_q \setminus \mathscr{B}_\epsilon(0)$. Since $V(\theta_0) \leq b$, there holds

$$V(\theta_{m+1}) \leq V(\theta_m) - \gamma \leq V(\theta_0) - (m+1)\gamma \leq b - (m+1)\gamma, \tag{4.23}$$

where the right-hand side after some $m > 0$ must become negative, contradicting the strict positivity of V over $\Xi_q \setminus \mathscr{B}_\epsilon(0)$ guaraneed by (4.15b). Therefore, the subsequence $\{\theta_m\}_\kappa$ of all iterates outside of $\mathscr{B}_\epsilon(0)$ cannot be infinite, and thus there is a $K > 0$ such that all elements of the sequence $\{\theta_k\}$ with $k \geq K$ are in the ball $\mathscr{B}_\epsilon(0)$, so $\lim_{k\to\infty} \theta_k = 0$. □

Remark 11. *The observations from Remark 9 and Remark 10 stated in the continuous-time setting also hold for the discrete-time setting.*

4.2.2 Robust optimization problem for finding Lyapunov function

Assume that the continuous-time or discrete-time Lyapunov function $V(\theta)$ is a linear combination of n_v basis functions:

$$V(\theta) = c_\mathrm{v}^T M(\theta), \tag{4.24}$$

where $c_\mathrm{v} \in \mathbb{R}^{n_\mathrm{v}}$ is a vector of Lyapunov weighting factors, and where $M : \mathbb{R}^{n_\theta} \to \mathbb{R}^{n_\mathrm{v}}$, $M(0) = 0$ is a vector of basis functions (for instance, a vector containing all monomials up to a certain degree). To make the following description applicable to both the continuous-time and the discrete-time case, the corresponding Lyapunov conditions (4.7) and (4.15) are stated in a unified way as

$$c_\mathrm{v}^T \bar{M}(\theta) \geq \|\theta\|_2^2, \quad \forall \theta \in \Xi, \tag{4.25a}$$

$$c_\mathrm{v}^T M(\theta) \geq \|\theta\|_2^2, \quad \forall \theta \in \Xi, \tag{4.25b}$$

where in the continuous-time case the $\bar{M}(\theta)$ has the form

$$\bar{M}(\theta) = -\nabla M(\theta)^T f_{cl}(\theta) \tag{4.26}$$

with $M(\theta)$ continuously differentiable, and in the discrete-time case

$$\bar{M}(\theta) = M(\theta) - M(\theta^+) \tag{4.27}$$

with $M(\theta)$ continuous. The problem of finding a Lyapunov function $V(\theta)$ will now be considered as a problem of finding weighting factors c_v such that conditions (4.25) are satisfied by the Lyapunov function (4.24). Moreover, the weighting factors c_v which are as much as possible in the interior of the set of all c_v for which (4.25) holds will be sought. This is achieved by an optimization problem which inscribes a ball of largest possible radius c_r and center at c_v into the set of all Lyapunov weighting factors for which (4.25) holds. By denoting the ball of radius c_r whose center is at c_v by

$$\mathscr{B}(c_\mathrm{v}, c_\mathrm{r}) = \{\tilde{c}_\mathrm{v} \in \mathbb{R}^{n_\mathrm{v}} \mid \|\tilde{c}_\mathrm{v} - c_\mathrm{v}\|_2 \leq c_\mathrm{r}\}, \tag{4.28}$$

the optimization problem inscribing a ball of largest radius into the set of Lyapunov weighting factors satisfying (4.25) takes the form:

$$\begin{aligned} \max_{c_\mathrm{v}, c_\mathrm{r}} \quad & c_\mathrm{r} \\ \text{s.t.} \quad & \tilde{c}_\mathrm{v}^T \bar{M}(\theta) \geq \|\theta\|_2^2, \qquad \forall \tilde{c}_\mathrm{v} \in \mathscr{B}(c_\mathrm{v}, c_\mathrm{r}), \ \ \forall \theta \in \Xi, \\ & \tilde{c}_\mathrm{v}^T M(\theta) \geq \|\theta\|_2^2, \qquad \forall \tilde{c}_\mathrm{v} \in \mathscr{B}(c_\mathrm{v}, c_\mathrm{r}), \ \ \forall \theta \in \Xi, \\ & c_\mathrm{r} \leq c_\mathrm{r,max}, \end{aligned} \tag{4.29}$$

where the third constraint with some fixed $c_\mathrm{r,max} > 0$ has been added to prevent the radius from going to infinity, which may happen if the interior of the set of all Lyapunov weighting factors for which (4.25) holds is not bounded.

It can be seen that the obtained optimization problem (4.29) involves robustness requirements with respect to all $\tilde{c}_\mathrm{v} \in \mathscr{B}(c_\mathrm{v}, c_\mathrm{r})$ and all $\theta \in \Xi$. These two robustness requirements will be addressed in the following two subsections to obtain a more easily solved optimization problem.

4.2.3 Application of Chebychev center technique

To address the robustness with respect to $\tilde{c}_\mathrm{v} \in \mathscr{B}(c_\mathrm{v}, c_\mathrm{r})$, the procedure known in the literature as finding a Chebyshev center of a set [15] will be applied. To do this, for some fixed $\theta \in \Xi$, the first and second constraint can be rewritten in the form

$$(c_\mathrm{v} + d)^T \bar{M}(\theta) \geq \|\theta\|_2^2, \ \ \forall d : \|d\|_2 \leq c_\mathrm{r}, \tag{4.30a}$$

$$(c_\mathrm{v} + d)^T M(\theta) \geq \|\theta\|_2^2, \ \ \forall d : \|d\|_2 \leq c_\mathrm{r}, \tag{4.30b}$$

which is equivalent to

$$\inf_{\|d\|_2 \le c_\mathrm{r}} \left\{ (c_\mathrm{v} + d)^T \bar{M}(\theta) \right\} \ge \|\theta\|_2^2, \tag{4.31a}$$

$$\inf_{\|d\|_2 \le c_\mathrm{r}} \left\{ (c_\mathrm{v} + d)^T M(\theta) \right\} \ge \|\theta\|_2^2, \tag{4.31b}$$

and results in the reformulation

$$c_\mathrm{v}^T \bar{M}(\theta) - c_\mathrm{r} \|\bar{M}(\theta)\|_2 \ge \|\theta\|_2^2, \tag{4.32a}$$

$$c_\mathrm{v}^T M(\theta) - c_\mathrm{r} \|M(\theta)\|_2 \ge \|\theta\|_2^2. \tag{4.32b}$$

The optimization problem (4.29) thus takes the form

$$\begin{aligned} \max_{c_\mathrm{v}, c_\mathrm{r}} \quad & c_\mathrm{r} \\ \text{s.t.} \quad & g_1(c_\mathrm{v}, c_\mathrm{r}, \theta) \ge 0, \ \ \forall \theta \in \Xi, \\ & g_2(c_\mathrm{v}, c_\mathrm{r}, \theta) \ge 0, \ \ \forall \theta \in \Xi, \\ & c_\mathrm{r} \le c_\mathrm{r,max}, \end{aligned} \tag{4.33}$$

where the functions $g_1(c_\mathrm{v}, c_\mathrm{r}, \theta)$ and $g_2(c_\mathrm{v}, c_\mathrm{r}, \theta)$ are introduced to make the notation lighter and are defined by

$$g_1 = c_\mathrm{v}^T \bar{M}(\theta) - c_\mathrm{r} \|\bar{M}(\theta)\|_2 - \|\theta\|_2^2, \tag{4.34a}$$

$$g_1 = c_\mathrm{v}^T M(\theta) - c_\mathrm{r} \|M(\theta)\|_2 - \|\theta\|_2^2, \tag{4.34b}$$

so that they represent the functions of the reformulation (4.32).

4.2.4 Application of scenario-based optimization technique

To address the robustness with respect to $\theta \in \Xi$, instead of seeking a solution that holds with perfect accuracy (i.e., holds for $\Lambda_\% = 100\%$ of the elements of Ξ), in order to facilitate computational tractability a solution with negligibly small approximation error will be sought (e.g., a solution that holds not for $\Lambda_\% = 100\%$, but for at least $\Lambda_\% = 99.99\%$ of the elements of Ξ), with the approximation error bounded and inversely proportional to the invested computation time. This can be achieved by application of the scenario approach whose theory provides a trade-off between the selected accuracy $\Lambda_\%$ and the required computation time (i.e., required number of scenarios), providing thus the computation time and accuracy trade-off which is standard in many fields of numerical mathematics (e.g., in numerical optimization). Due to the imperfect accuracy which can be achieved by this computational procedure (i.e., inability to achieve $\Lambda_\% = 100\%$ with finite computation time), the solutions generated by this approach should be referred to as numerical Lyapunov estimates for the controller synthesis method.

To apply the scenario approach [20, 21], it should be noted that for every fixed $\theta \in \Xi$ the functions $g_1(c_\mathrm{v}, c_\mathrm{r}, \theta)$ and $g_2(c_\mathrm{v}, c_\mathrm{r}, \theta)$ are linear in the decision vector $(c_\mathrm{v}, c_\mathrm{r})$. By assuming a

uniform probability distribution of θ over the compact set Ξ and by extracting N_{sc} independent samples $\{\theta_i\}_{i=1}^{N_{\text{sc}}}$, the optimization problem (4.33) takes the form

$$\begin{aligned} \max_{c_{\text{v}},c_{\text{r}}} \quad & c_{\text{r}} \\ \text{s.t.} \quad & g_1(c_{\text{v}},c_{\text{r}},\theta_i) \geq 0, \quad i=1,\ldots,N_{\text{sc}}, \\ & g_2(c_{\text{v}},c_{\text{r}},\theta_i) \geq 0, \quad i=1,\ldots,N_{\text{sc}}, \\ & c_{\text{r}} \leq c_{\text{r,max}}, \end{aligned} \tag{4.35}$$

which is an LP optimization problem and thus can be solved efficiently even for a large number of scenarios N_{sc}. As will be stated more formally in the theorem that follows, even though the problem (4.35) contains only a finite number of instances of θ from Ξ, the scenario approach theory allows one to quantify the percentage $\Lambda_{\%}$ of the elements θ from Ξ for which the solution of (4.35) is feasible (i.e., the solution of (4.35) holds not only for the used $\{\theta_i\}_{i=1}^{N_{\text{sc}}}$, but for $\Lambda_{\%}$ elements of the set Ξ), and the theory also quantifies that the stated $\Lambda_{\%}$ is incorrect only with some user-controlled probability σ. The scenario approach theory therefore provides a relationship between the number of scenarios N_{sc} and the accuracy $\Lambda_{\%}$ with which the problem (4.35) approximates the original form (4.33), allowing thereby a trade-off between the computational time (i.e., the number of used scenarios N_{sc}) and the accuracy of the obtained solution (i.e., the achieved $\Lambda_{\%}$). As will be demonstrated in Section 4.4, in case of the problem sizes considered in the application examples, the scalability of linear programming allows numbers of scenarios N_{sc} with which practically convenient values of $\Lambda_{\%}$ and σ (e.g., accuracy $\Lambda_{\%} = 99.99\%$ and probability $\sigma = 10^{-7}$) are achievable. Figure 4.1 illustrates the scenario approach concept and the Chebychev center technique.

To apply the scenario approach theory [21], the solution of the problem (4.35) should be verified to exist and be unique ([21], Assumption 1). To avoid the technical difficulties concerning this requirement, as indicated in Section 3.1 of [20], the solution existence can be ensured by adding a compact constraint set $\|(c_{\text{v}},c_{\text{r}})\|_2 \leq 10^{99}$ (whose presence guarantees the solution existence by the Weierstrass theorem), and the solution uniqueness can always be ensured by applying a tie-break rule.

The relationship between the number of scenarios N_{sc}, accuracy $\Lambda_{\%}$ and confidence σ is given by Theorem 1 in [21], which is rephrased below to a form tailored to the present setting.

Theorem 3. *Let N_{sc} be the number of uniformly sampled scenarios $\{\theta_i\}_{i=1}^{N_{\text{sc}}}$ from Ξ with which the robust optimization problem* (4.33) *is converted into* (4.35)*. Then, it holds that the solution of* (4.35) *is feasible for at least $\Lambda_{\%}$ percent of the elements $\theta \in \Xi$, with the probability σ that it is not the case upper-bounded by the expression*

$$\sigma \leq \frac{1}{100^{N_{\text{sc}}}} \sum_{m=0}^{n_{\text{v}}} \binom{N_{\text{sc}}}{m} \left(100-\Lambda_{\%}\right)^{m} \left(\Lambda_{\%}\right)^{N_{\text{sc}}-m}, \tag{4.36}$$

where n_{v} is the number of basis functions used for the Lyapunov function in (4.24).

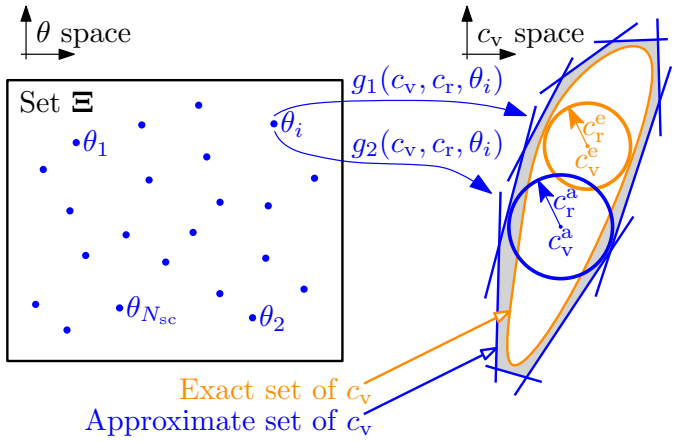


Figure 4.1 – A simplistic illustration of the application of the scenario approach. The optimization problem (4.33) involves mapping of the set Ξ by the functions g_1 and g_2 into the orange set represented in the space of c_v, which as such results in the inscribed ball $\mathscr{B}(c_\mathrm{v}^\mathrm{e}, c_\mathrm{r}^\mathrm{e})$ that is the exact solution of (4.33). The scenario approach (4.35) considers only N_sc samples of $\theta \in \Xi$, denoted $\{\theta_i\}_{i=1}^{N_\mathrm{sc}}$, which are mapped through g_1 and g_2 into the blue polytopic set that makes an outer approximation of the exact orange set, thus causing an approximate solution $\mathscr{B}(c_\mathrm{v}^\mathrm{a}, c_\mathrm{r}^\mathrm{a})$. The difference between the two sets is colored in gray. The scenario approach theory states that only those elements which do not belong to a $\Lambda_\%$ percentage of Ξ could possibly generate some halfspaces which would not fully contain the ball $\mathscr{B}(c_\mathrm{v}^\mathrm{a}, c_\mathrm{r}^\mathrm{a})$ in them, a statement whose probability of not being true is σ.

The practical application of the above theorem can be performed by selecting an accuracy $\Lambda_\%$ and an upper bound $\hat{\sigma}$ for the right-hand side in (4.36):

$$\frac{1}{100^{N_\mathrm{sc}}} \sum_{m=0}^{n_\mathrm{v}} \binom{N_\mathrm{sc}}{m} (100 - \Lambda_\%)^m (\Lambda_\%)^{N_\mathrm{sc}-m} \leq \hat{\sigma}, \tag{4.37}$$

and then tackling the above inequality by the bisection method on N_sc (due to the decreasing monotonicity in N_sc of the left-hand side in (4.37)) in order to obtain the smallest number of scenarios N_sc so that the inequality (4.37) holds, resulting also in some σ probability that satisfies $\sigma \leq \hat{\sigma}$. Another way to obtain an appropriate number of scenarios N_sc in (4.37) given $\Lambda_\%$ and $\hat{\sigma}$ is by using the lower bound (see [20], Section 5):

$$N_\mathrm{sc} \geq \frac{200}{100 - \Lambda_\%} \left[\ln\left(\frac{1}{\hat{\sigma}}\right) + n_\mathrm{v} \right], \tag{4.38}$$

which will be used for obtaining the numbers of scenarios in the application examples of Section 4.4.

4.3 Controller Synthesis

The controller synthesis can be formulated as in the Chapter 2 through an optimization problem of the form

$$\begin{aligned} &\text{min.} \quad \mathrm{P}(\eta) + \delta_{st}(\eta) \\ &\text{s.t.} \quad \eta \in \mathbf{D}, \end{aligned} \tag{4.39}$$

in which $\mathrm{P}(\eta)$ is a user-specified performance criteria for the closed-loop system with control parameters η, the $\delta_{st}(\eta)$ is a function indicating that a Lyapunov function is estimated to exist by (4.35) (i.e., there is a solution with positive radius c_{r}):

$$\delta_{st}(\eta) = \begin{cases} 0, & \text{for } \eta \text{ with positive } c_{\mathrm{r}} \text{ from (4.35)}, \\ +\infty, & \text{otherwise}, \end{cases} \tag{4.40}$$

and the set $\mathbf{D}$ models some basic requirements on the tuning parameters η (e.g., a requirement that the Hessian in the cost function of a QP-based controller should be symmetric positive definite).

The solving of optimization problem (4.39) is approached in two phases, similarly as in the method previously described in Chapter 2 which was based on SOS stability certification. The first phase searches for control parameters η feasible in (4.39) (i.e., for parameters η for which a Lyapunov function can be generated by (4.35)). The second phase takes the stabilizing tuning parameters generated by the first phase as initial conditions which are providing an indication about the location of a stabilizing region, and then starting from that data further explores for improvement of the performance criteria $\mathrm{P}(\eta)$ by means of Bayesian optimization and its data exploitation property. In comparison to the approach in Chapter 2, the first phase of the method that will be described in this section does not require the introduction of a slack polynomial into the Lyapunov decrease condition (4.25a) whose minimization provides stabilizing tuning parameters. Instead, the stabilizing tuning parameters are searched for by maximizing the radius of the inscribed ball from (4.35) as a function of the tuning parameters η, as discussed in the following subsection.

4.3.1 Phase one - Search for stabilizing controllers

Denote with $(c_{\mathrm{v}}^*, c_{\mathrm{r}}^*)$ the optimal solution of (4.35) obtained for some fixed tuning vector η. To emphasize the dependence of this optimal solution on the tuning parameters η, the notation $(c_{\mathrm{v}}^*(\eta), c_{\mathrm{r}}^*(\eta))$ will be used in what follows. In case of the tuning parameters η for which there can be generated a Lyapunov function by (4.35), the value of the radius is nonnegative (i.e., $c_{\mathrm{r}}^*(\eta) \geq 0$). Otherwise, in the case where η is such that there is no Lyapunov function that can be generated by (4.35), a feasible optimal solution $(c_{\mathrm{v}}^*(\eta), c_{\mathrm{r}}^*(\eta))$ is still obtained, but it is characterized by a negative value of the radius c_{r}^* (i.e., $c_{\mathrm{r}}^*(\eta) < 0$). The magnitude $|c_{\mathrm{r}}^*(\eta)|$ of this negative radius provides indication about how far the selected tuning η is from being able to

have a Lyapunov function generated.

The previous observations lead to the following optimization problem for the search of stabilizing tuning parameters η:

$$\begin{aligned} &\text{max.} \quad \mathrm{I}_r\left(c_{\mathrm{r}}^*(\eta)\right) \\ &\text{s.t.} \quad \eta \in \mathbf{D}, \end{aligned} \tag{4.41}$$

where the set $\mathbf{D}$ models some basic design requirements on η as in (4.39), and the function $\mathrm{I}_r : \mathbb{R} \to \mathbb{R}$ introduces saturation for the positive values of its argument:

$$\mathrm{I}_r(\gamma) = \begin{cases} 0, & \text{for } \gamma \geq 0, \\ \gamma, & \text{otherwise.} \end{cases} \tag{4.42}$$

The reason for introducing the saturation function $\mathrm{I}_r(\gamma)$ is that it makes the cost function in (4.41) equal to zero for all tuning parameters η for which a Lyapunov function from (4.35) can be generated. As such, it eliminates bias towards the parameters η with larger radius values $c_{\mathrm{r}}^*(\eta)$ and allows a uniform exploration of the region with stabilizing parameters η when Bayesian Optimization is applied to (4.41). In the application examples of this chapter, the optimization problem (4.41) for obtaining stabilizing tuning parameters will be as well addressed by using Bayesian optimization [17] as in Chapter 2, where in Section 2.3.2 an overview of Bayesian optimization algorithmic concept is given.

4.3.2 Phase two - Optimization of performance criteria

The second phase takes the tuning parameters generated by the first phase for which it is estimated that Lyapunov functions exist, and uses these tuning parameters as the initial condition for solving the synthesis optimization problem (4.39). The second phase is identical to the second phase of the SOS-based method previously described in Section 2.3.3, meaning that the optimization problem (4.39) will be addressed by Bayesian optimization where a certain number of tuning parameters η found by phase one will be used to provide some information about the location of the stabilizing region in the space of tuning parameters, which would then be a region of focus for further investigation while minimizing the performance criteria $\mathrm{P}(\eta)$.

As in Chapter 2, there is a great amount of flexibility in the choice of the cost term $\mathrm{P}(\eta)$ in (4.39), as it is allowed to be any performance criteria which can be evaluated for a fixed vector of tuning parameters η. For example, the performance criteria described in Section 2.3.3 would in the discrete-time setting involve an approximate evaluation of the integral of the infinite horizon trajectory cost over some set $\mathbf{W}$:

$$\mathrm{P}(\eta) = \int_{\mathbf{W}} C_{\infty,\eta}(x)\,dx, \quad C_{\infty,\eta}(x) = \sum_{k=0}^{\infty} l(x_k, u_k), \tag{4.43}$$

where $C_{\infty,\eta}(x)$ is the infinite horizon trajectory cost obtained with controller η when starting from the state x, the $l(x,u)$ is some stage cost, and x_k and u_k are the state and input values at the k-th step starting from $x_0 = x$. The cost $\mathrm{P}(\eta)$ of (4.43) can be evaluated approximately by using Monte Carlo (MC) approximation for the integral and finite horizon approximations for the trajectory costs:

$$\mathrm{P}(\eta) = \sum_{j=1}^{N_{mc}} \tilde{C}_{N_{st},\eta}(x_j), \;\; \tilde{C}_{N_{st},\eta}(x) = \sum_{k=0}^{N_{st}} l(x_k, u_k), \tag{4.44}$$

where $\tilde{C}_{N_{st},\eta}(x)$ is the finite horizon trajectory cost involving N_{st} simulation steps from the initial state x, and the N_{mc} is the number of samples from the set $\mathbf{W}$ in the MC approximation of the integral.

4.4 Application Examples

This section demonstrates the method on examples which involve the synthesis of easily computable controllers for the constrained control of nonlinear systems. The first example (given in Section 4.4.1) demonstrates the continuous-time version of the method by doing a synthesis of a soft-constrained speed controller of a (bilinear) permanent magnet synchronous machine (PMSM) model where the optimization-based controller involves an early-terminated extrapolated gradient-projection optimization algorithm with approximate projections. The second example (given in Section 4.4.2) demonstrates the discrete-time version of the method by synthesizing a position controller of a gyroscope whose model is characterized by nonlinear system dynamics and input constraints. The LP optimization problems for generation of Lyapunov functions are implemented by using Yalmip [48] as a modelling tool together with MOSEK [5] as LP solver, and the Bayesian optimization is applied by using Matlab's Statistics and Machine Learning Toolbox [49].

4.4.1 Controller with early-terminated extrapolated gradient projection algorithm for soft-constrained control of a bilinear plant

This section provides an application example in the continuous-time setting by synthesizing an optimization-based controller with an early terminated optimization algorithm for speed control of a (bilinear) permanent magnet synchronous machine (PMSM) with soft-constrained stator current magnitude. Since the synthesis of the control policy (defined by the optimization-based controller's structure) is done in continuous-time, the subsequent practical application of the obtained controller should be done by using a small control period which approximates a continuous-time execution.

The optimization algorithm used to address the controller's optimization problem is a gradient-projection method with extrapolations, implemented with a constant stepsize and constant extrapolation factor, and involves approximate projections on the constraint set. It is inter-

esting to observe that this optimization algorithm does not have convergence guarantees, as in order to attain the $O(1/k^2)$ iteration complexity (see, e.g., [52] or [13]) one requires exact projections on the constraint set (instead of approximate ones used here) as well as a complex updating scheme for the extrapolation factor (instead of using a constant one as here). The ability of the synthesis method to consider a heuristic early-terminated optimization algorithm originates from the fact that the method's focus is only on the obtained control policy and the consequent closed-loop behaviour, which can be shaped by tuning not only the coefficients of the optimization problem data (i.e., the coefficients in matrices and vectors describing the cost function and constraint set) but also the parameters of the optimization algorithm (i.e., stepsize and extrapolation factor), as will be done in what follows. Thus, the convergence guarantee of the optimization algorithm is not significant, as the actual control policy and the closed-loop behavior are directly shaped by the method.

In comparison to the MPC scheme for PMSM developed in [24], the selected structure of the optimization-based controller of this section allows incorporation of the soft constraint effect in a manner that can bring a large reduction of the number of decision variables in comparison to the optimization problem obtained in [24]. Furthermore, the controller synthesis of this section directly deals with the bilinear model of the system, thus circumventing the need for using a linear prediction model like in [24] which is valid only at nominal (or some other fixed and in advance chosen) rotational speed. In addition, the synthesised controller of this section removes the outer speed control loop based on a PI controller, and as indicated earlier also involves the aforementioned early-terminated optimization algorithm which as such has deterministic computational time. In comparison to the example in Section 2.4 where a QP controller for PMSM was synthesized by using SOS-based method, the SOCP controller of this section includes soft constraints, involves presence of a non-zero load torque, and also involves the early-terminated first order optimization algorithm, which are the elements whose adressing would not be possible by the approach of Chapter 2 due to the scalability limitations arising form the presence of SOS programming.

The PMSM involved in this example is the same as the one considered in Section 2.4, and the description of its continuous-time model and state/input vectors is repeated in this paragraph to allow an easier following of the further developments. The continuous-time model of the two-pole PMSM in the dq reference frame fixed to the rotor (see, e.g., [44]) has the form

$$\frac{dI_d(t)}{dt} = -\frac{R_s}{L_s} I_d(t) + \Omega_r(t) I_q(t) + \frac{1}{L_s} U_d(t), \tag{4.45a}$$

$$\frac{dI_q(t)}{dt} = -\frac{R_s}{L_s} I_q(t) - \left(I_d(t) + \frac{\Phi_0}{L_s} \right) \Omega_r(t) + \frac{1}{L_s} U_q(t), \tag{4.45b}$$

$$\frac{d\Omega_r(t)}{dt} = \frac{K_t}{J} I_q(t) - \frac{1}{J} \Gamma_{_L}(t), \tag{4.45c}$$

where $I_d(t)$ and $I_q(t)$ are the d and q component of the stator current vector $I(t) = [I_d(t), I_q(t)]^T$, $\Omega_r(t)$ is the rotational speed of the rotor, $U_d(t)$ and $U_q(t)$ are the d and q component of the input voltage vector $U(t) = [U_d(t), U_q(t)]^T$, $\Gamma_{_L}(t)$ is the load torque, the parameter R_s is stator

resistance, L_s stator inductance, Φ_0 the flux from the rotor's permanent magnet, K_t torque coefficient and J the rotational inertia of the rotor. The Table 2.1 contains the parameters of PMSM from [24] which will be used in this numerical example. The system has an input constraint $\|U(t)\|_2 \leq U_{nom}$ which concerns the magnitude of the input voltage vector. Due to the thermal inertia of the machine, the stator current vector $I(t)$ is allowed to make temporary violations of the constraint $\|I(t)\|_2 \leq I_{nom}$ during transients and will thus be addressed as a soft-constraint. By using the base values in Table 2.1, the following normalized model is obtained:

$$\frac{di_d(\tau)}{d\tau} = -\frac{r_s}{l_s} i_d(\tau) + \omega_r(\tau)\, i_q(\tau) + \frac{1}{l_s} u_d(\tau), \tag{4.46a}$$

$$\frac{di_q(\tau)}{d\tau} = -\frac{r_s}{l_s} i_q(\tau) - \left(i_d(\tau) + \frac{\phi_0}{l_s}\right)\omega_r(\tau) + \frac{1}{l_s} u_q(\tau), \tag{4.46b}$$

$$\frac{d\omega_r(\tau)}{d\tau} = \mathcal{K}_t i_q(\tau) - \frac{\Gamma_b}{J\omega_b^2}\gamma_{_L}(\tau), \tag{4.46c}$$

where $\tau = \omega_b t$ is per-unit time, $i_d(\tau) = I_d(t)/I_b$, $i_q(\tau) = I_q(t)/I_b$ are per-unit stator current components, $\omega_r(\tau) = \Omega_r(t)/\omega_b$ per-unit rotational speed, $u_d(\tau) = U_d(t)/U_b$, $u_q(\tau) = U_q(t)/U_b$ per-unit input voltage components, $\gamma_{_L}(\tau) = \Gamma_{_L}(t)/\Gamma_b$ per-unit load torque, and the per-unit parameters appearing in the model are as defined in Table 2.1. The input constraint for the per-unit model takes the form $\|u(\tau)\|_2 \leq 1$, and the constraint on the current which can be temporarily violated during transients $\|i(\tau)\|_2 \leq 1$. The corresponding state vector is $x = [i_d, i_q, \omega_r]^T$ and the input vector $u = [u_d, u_q]^T$. It can be seen that the plant model is bilinear as it involves products of state variables.

For the purpose of tracking a constant speed reference r by using the delta-space formulation, the steady-state target operating point (x_s, u_s) of the form

$$x_s = \begin{bmatrix} i_{ds} \\ i_{qs} \\ \omega_{rs} \end{bmatrix}, \quad u_s = \begin{bmatrix} u_{ds} \\ u_{qs} \end{bmatrix}, \tag{4.47}$$

at which the rotational speed is equal to r is to be computed. In case involving a constant load torque $\gamma_{_L}$ (which is assumed to be provided to the controller by an estimator), the components of the steady-state target (x_s, u_s) are $i_{ds} = 0$, $i_{qs} = \frac{\Gamma_b}{\mathcal{K}_t J\omega_b^2}$, $\omega_{rs} = r$, $u_{ds} = -l_s\omega_{rs}i_{qs}$, $u_{qs} = r_s i_{qs} + \phi_0\omega_{rs}$. Consideration of a load torque of some different shape (e.g., a $\gamma_{_L}$ which is a polynomial function of the rotational speed) is also possible and reflects itself on the expression for the steady-state target (4.47).

The SOCP-based controller will be synthesized so that the SOCP optimization problem takes as its inputs the state x, the target u_s and the deviation from the steady-state target $\Delta x = x - x_s$, and as its output (its solution decision vector) provides the deviation Δu from the steady-state target u_s (i.e., the input signal to the PMSM is $u = u_s + \Delta u$). For the purpose of incorporating the soft constraint on the current magnitude $\|i(t)\|_2 \leq 1$, denote the current value predicted

$T_s = 100\,\mu s$ into the future by

$$i_{sc}^{+}(z;x,u_s) = \begin{bmatrix} a_{11}i_d + a_{12}\omega_r i_q + b_1(u_{sd} + z_1) \\ a_{21}i_q - a_{22}\omega_r i_d - a_{23}\omega_r + b_2(u_{sq} + z_2) \end{bmatrix} \tag{4.48}$$

which is obtained from (4.46a)-(4.46b) by applying forward Euler's method and involves $a_{11} = a_{21} = (1 - T_s\omega_b r_s/l_s)$, $a_{12} = a_{22} = T_s\omega_b$, $b_1 = T_s\omega_b/l_s$, $a_{23} = T_s\omega_b\phi_0/l_s$, $b_2 = T_s\omega_b/l_s$. In what follows, the $i_{sc}^{+}(z;x,u_s)$ will be denoted by $i_{sc}^{+}(z)$ to make the notation lighter. The form of the SOCP is selected to be

$$\begin{aligned} \underset{\xi=[z^T,\delta]^T}{\text{minimize}} \quad & \tfrac{1}{2}z^T Hz + \Delta x^T Fz + m\delta^2 \\ \text{subject to} \quad & \|u_s + z\|_2 \le 1, \\ & \|i_{sc}^{+}(z)\|_2 \le (1+\delta), \end{aligned} \tag{4.49}$$

where $\xi = [z^T,\delta]^T \in \mathbb{R}^3$ is the decision vector whose components $z \in \mathbb{R}^2$ correspond to the input deviation Δu to be applied to the PMSM and $\delta \in \mathbb{R}$ corresponds to a slack variable, $H \in \mathbb{R}^{2\times 2}$ is a symmetric positive definite matrix, $F \in \mathbb{R}^{3\times 2}$, and $m \in \mathbb{R}$ is the slack penalty. The first constraint of (4.49) represents the input constraint $\|u\|_2 \le 1$, while the second one is supposed to introduce a soft constrained effect on $\|i\|_2 \le 1$. It is interesting to notice that the input parameters are entering the optimization problem (4.49) in a nonlinear manner through the function $i_{sc}^{+}(z)$ since it involves products of the components of x (products of the rotational speed ω_r with the current components i_d and i_q), as can be seen in (4.48). The elements of the H and F matrix, as well as the scalar m, will be considered as controller's tuning parameters and thus contained in η.

The optimization problem (4.49) is addressed by approximate extrapolated gradient-projection algorithm with early termination after $N_{\text{it}} = 10$ iterations. The pseudocode of the algorithm is given in Algorithm 1. Each iteration k of the algorithm involves three substeps. The first substep represents extrapolation:

$$\xi_{e,k} = \xi_k + \beta(\xi_k - \xi_{k-1}), \tag{4.50}$$

where $\xi_{e,k}$ is the extrapolated vector, and β is a (constant) extrapolation factor which will be included as one of the tuning parameters in η. In the first iteration (i.e., for $k = 0$), the algorithm will be initialized with $\xi_0 = \xi_{-1} = 0$. The second substep of each iteration is a gradient step from the extrapolated vector $\xi_{e,k}$:

$$\tilde{\xi}_k = \xi_{e,k} - \alpha\nabla_\xi f(\xi_{e,k}), \tag{4.51}$$

where $f(\xi)$ denotes the cost function of the SOCP optimization problem (4.49), and α is a constant stepsize which will as well be considered as a tuning parameter and thus contained in η. The third substep of iteration represents the approximate projeciton on the feasible set of (4.49). It is done by performing alternating projections for $N_{\text{pr}} = 5$ times on the first and

the second constraint in (4.49), with one additional projection on the first constraint at the end in order to ensure the satisfaction of the input constraint. Each of these projections is a projection on a second-order cone set, which as such has a closed-form expression (see, e.g., [6], Theorem 3.6.6).

Algorithm 1 Approximate gradient-projection with constant stepsize and extrapolation factor for SOCP (4.49).

Require: N_{it} , N_{pr} , α , β , $\xi_{-1} = \xi_0 = [z_0^T, \delta_0]^T$;
for $k := 0$ to N_{it} **do**
extrapolation:
  $\xi_{e,k} := \xi_k + \beta(\xi_k - \xi_{k-1})$;
gradient step:
  $\tilde{\xi}_k := \xi_{e,k} - \alpha \nabla_\xi f(\xi_{e,k})$;
approximate projection:
  do N_{pr} *alternating projections on the two constraints*:
  for $j := 1$ to N_{pr} **do**
    $\tilde{\xi}_k :=$ projection of $\tilde{\xi}_k$ on $\|u_s + z\|_2 \leq 1$;
    $\tilde{\xi}_k :=$ projection of $\tilde{\xi}_k$ on $\|i_{sc}^+(z)\|_2 \leq (1+\delta)$;
  end for
  do one more projection on the first constraint:
  $\tilde{\xi}_k :=$ projection of $\tilde{\xi}_k$ on $\|u_s + z\|_2 \leq 1$;
store the obtained ξ_k *for the next iteration*:
  $\xi_k := \tilde{\xi}_k$;
end for
return $\xi_{N_{\text{it}}}$.

The controller's tuning parameters (the elements of vector η) thus consist of the coeficients of the H and F matrix, the scalar m in the SOCP cost function, and the tuning parameters of the optimization algorithm (the stepsize α and the extrapolation factor β), resulting in $\eta \in \mathbb{R}^{12}$ (due to the symmetry of H).

For the purpose of tracking a constant speed reference, the Lyapunov conditions (4.7) are formulated in delta space:

$$-\nabla_{\Delta x} V(\Delta x, r, \gamma_L)^T f_{cl}(\Delta x, r, \gamma_L) \geq \|\Delta x\|_2^2, \tag{4.52a}$$

$$V(\Delta x, r, \gamma_L) \geq \|\Delta x\|_2^2, \tag{4.52b}$$

where it can be seen that the Lyapunov function is specified to also depend on the speed reference r and the load torque γ_L, and the $f_{cl}(\Delta x, r, \gamma_L)$ is the right-hand side of (4.46) with the control input u corresponding to the described control policy.

The set Ξ over which the Lyapunov conditions are enforced is specified to involve bounds on the speed reference $|r| \leq r_{\max}$ with $r_{\max} = 1$, bounds on the current magnitude $\|i\|_2 \leq i_{\max}$ with $i_{\max} = 2$, bounds on the rotational speed $|\omega_r| \leq \omega_{\max}$ with $\omega_{\max} = 1.5$, and bounds on the load torque $|\gamma_L| \leq \gamma_{\max}$ with $\gamma_{\max} = 1$. The local set Ξ is thus defined by $\Xi = \{x, r, \gamma_L \,|\, \psi \geq 0\}$

where the function ψ is

$$\psi = \begin{bmatrix} r + r_{\max} \\ -r + r_{\max} \\ i_{\max} - \|i\|_2 \\ \omega_r + \omega_{\max} \\ -\omega_r + \omega_{\max} \\ \gamma_L + \gamma_{\max} \\ -\gamma_L + \gamma_{\max} \end{bmatrix}. \tag{4.53}$$

The control synthesis is run with polynomial Lyapunov function $V(\Delta x, r, \gamma_L)$ where the vector of basis functions $M(\Delta x, r, \gamma_L)$ contains all monomials of degree two, except the monomials involving only combinations of r and γ_L which are constant and are thus excluded. Such choice of the vector of basis functions $M(\Delta x, r, \gamma_L)$ in (4.24) results in $c_{\mathrm{v}} \in \mathbb{R}^{12}$ (i.e., $n_{\mathrm{v}} = 12$). The search ranges for tuning parameters in η are chosen to be $[-1,1]$ for the elements of H and F matrices, $[0,1]$ for the stepsize α and extrapolation β of the optimization algorithm and $[1,3]$ for the slack weighting m, which are all together with the positive-definiteness constraint for the matrix H embedded into the set $\mathbf{D}$ involved in (4.41) and (4.39). The accuracy of the Lyapunov function computation in (4.35) is selected to $\Lambda_{\%} = 99.99\%$, which together with $\hat{\sigma} = 10^{-7}$ results in 562362 scenarios in (4.35).

After 150 Bayesian optimization iterations applied to (4.41) as a part of phase one, 51 tuning parameters η in the set $\{\eta \mid \mathrm{I}_p(\eta) = 0, \eta \in \mathbf{D}\}$ were obtained, with an average time per Bayesian optimization iteration of about 4.9 minutes. These parameters were then used as initial points in a total of 300 Bayesian optimization iterations applied to the performance optimization problem (4.39), with the average time per iteration of about 5.3 minutes. The performance criteria used was the MC approximation of the integral of trajectory costs, as given in (2.19). For it, the trajectories were simulated using $N_{st} = 1000$ steps of a discrete-time model obtained by the forward Euler method with stepsize of $100\mu s$, and N_{mc} was selected to $N_{mc} = 180$ which was spread equally to the cases involving all combinations of the reference values $r = 0$, $r = 0.8$ and $r = -0.8$ and the load-torque values $\gamma_L = 0.6\gamma_{\max}$, $\gamma_L = -0.6\gamma_{\max}$ and $\gamma_L = 0$. The stage cost was selected to be

$$l(x_k, u_k) = \begin{cases} \tilde{l}(x_k, u_k), & \text{for } \|i_k\| \le 1, \\ \tilde{l}(x_k, u_k) + q_{sc}\|i_k\|_2^2, & \text{otherwise,} \end{cases}$$

where the term

$$\tilde{l}(x_k, u_k) = \Delta x_k^T Q \Delta x_k + (u_k - u_{k-1})^T R (u_k - u_{k-1}),$$

penalizes the distance of the state x_k from the target x_s as well as the difference of the input vector at the two consecutive time instants, and the term $q_{sc}\|i_k\|_2^2$ introduces the soft-constraint effect into the performance metric. The $Q \in \mathbb{R}^{3\times 3}$ is selected to be a diagonal matrix with 1,

0 and 1 on its diagonal, the $R \in \mathbb{R}^{2\times 2}$ is diagonal with diagonal elements selected to 1, and the soft-constraint weighting q_{sc} was selected to 1. It can be seen that by the choice of Q, the current component i_q which produces the electromagnetic torque of the PMSM is left without any penalty, which is fine since the soft constraint will moderate the current magnitude. The set $\mathbf{W}$ used for initial states in (4.44) was selected identical to the local set Ξ specified by (4.53).

A slice of the control policy with the obtained H, F, m, α and β is represented on Fig. 4.2 with ω, r and γ_L fixed to $\omega = 0$, $r = 1$, $\gamma_L = 0.2$. The controller was tested in simulation by applying it with the sampling time $T_s = 100\mu s$ and starting it from many random initial points. Fig. 4.3 shows the state trajectory, the Lyapunov function values and the soft-constrained current vectors $[i_d, i_q]^T$ obtained with speed reference $r = 1$, load torque $\gamma_L = 0.2$ and a randomly generated initial state. It can be seen that the stator current vector involves a soft-constraint effect for the vector magnitudes $\|i\|_2 > 1$.

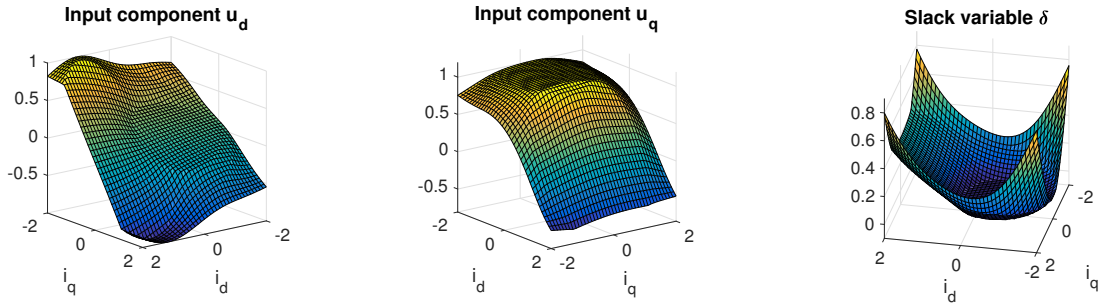


Figure 4.2 – The plant inputs $u = [u_d, u_q]^T$ of the obtained control policy and the corresponding slack δ values. The plotted slice of the control law involves ω, r, γ_L fixed to $\omega = 0$, $r = 1$, $\gamma_L = 0.2$, and the plots are generated by using a gridding of the i_d-i_q space involving 41 sampling points per each axis.

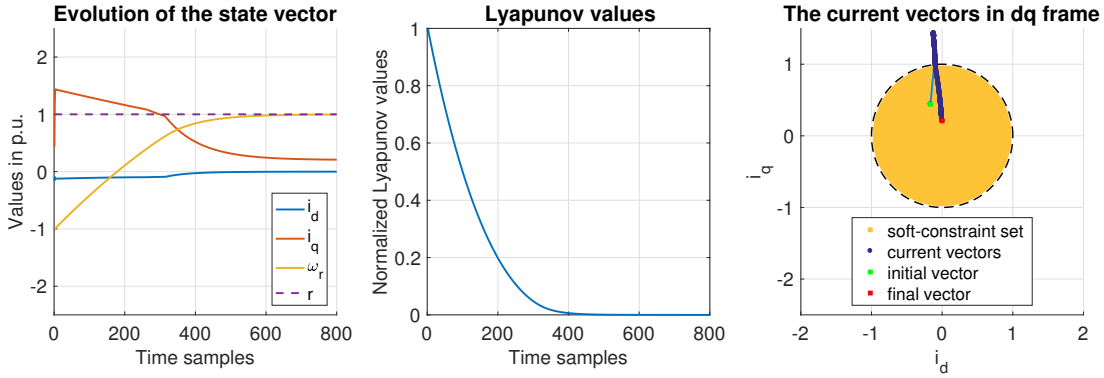


Figure 4.3 – Evolution of the state vector x starting from the initial $x_0 = [-0.17, 0.44, -1]^T$ for the reference $r = 1$ and load-torque $\gamma_L = 0.2$, the corresponding Lyapunov values along the state vector trajectory, and the current vectors $i = [i_d, i_q]^T$ together with the circle representing the soft-constraint.

Figure 4.4 – The Quanser 3DOF Gyroscope used in the experiments.

4.4.2 Cascaded linear controller for input-constrained control of a nonlinear plant

This section demonstrates the described method in a discrete-time setting on an example involving position control of a rotating disk of the gyroscope that was used in example of Section 3.4. The nonlinear dynamics of the gyroscope will be addressed directly, without introducing any linearizations. The goal is to perform input constrained nonlinear control that makes the two angles describing the rotating disk's position (α and β, as described below) go to zero (i.e., $\alpha \to 0$, $\beta \to 0$). Due to the symmetry of the system, the obtained controller would also allow reference tracking of the β angle by shifting of the origin of the state-space.

The gyroscope considered in the example is the Quanser 3DOF Gyroscope that was used in Section 3.4. It is represented again in Fig. 4.4 to allow an easier following of the description in this paragraph. The disk (whose position should be controlled) rotates on a blue ring which also carries a motor used to keep the disk's rotational speed constant. The blue ring carrying the disk is then connected to the outer red ring which as well contains a motor that can apply torque on the blue ring. The red ring is furthermore carried by a grey frame whose position is fixed in the experiments of this section and which also includes a motor that can apply torque on the red ring. In what follows, the angular position of the blue ring relative to the red ring will be denoted by α, and the angular position of the red ring relative to the grey frame by β. Furthermore, the torque applied on the blue ring will be denoted with M_α and the torque applied on the red ring with M_β. Since the position of the grey frame is fixed, one degree of

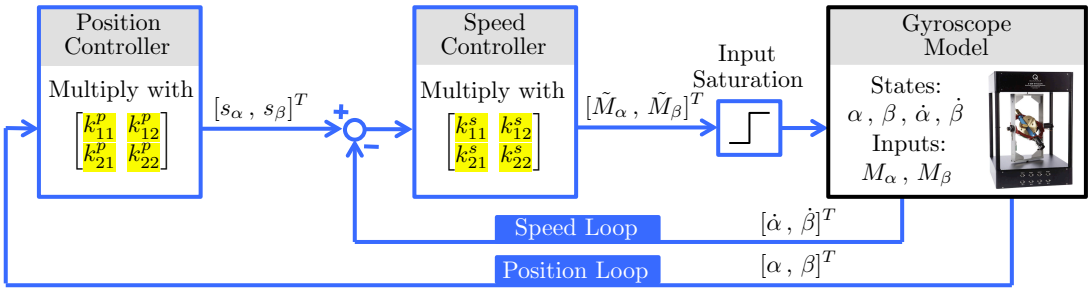


Figure 4.5 – Block diagram of the cascaded linear controller with input saturations (also described in Algorithm 2) which is used for gyroscope position control. The tuning parameters (the components of η) are the coefficients k^p_{ij} and k^s_{ij} with $i, j \in \{1,2\}$ which are highlighted in yellow in the figure. The input saturation block is saturating the components of the vector $[\tilde{M}_\alpha, \tilde{M}_\beta]^T$ at its input according to the expression (4.55).

Table 4.1 – Parameters of the gyroscope model.

Parameter	*Value*	*Unit*
J_x	0.0074	$kg\,m^2$
J_y	0.0026	$kg\,m^2$
J_z	0.0056	$kg\,m^2$
J^d_x	0.0056	$kg\,m^2$
J^r_Z	0.0286	$kg\,m^2$

freedom of the gyroscope is removed, thus resulting in a gyroscope under consideration which has two degrees of freedom (2DOF).

The dynamic model of 2DOF gyroscope can be derived by first principles modelling (see, e.g., [22]) and takes the form

$$J_y\ddot{\alpha} - J^d_x\,\omega\,\dot{\beta}\cos(\alpha) + (J_z - J_x)\dot{\beta}^2\sin(\alpha)\cos(\alpha) = M_\alpha, \tag{4.54a}$$

$$\left(J^r_Z + J_z\cos^2(\alpha) + J_x\sin^2(\alpha)\right)\ddot{\beta} + J^d_x\,\omega\,\dot{\alpha}\cos(\alpha) + 2\,(J_x - J_z)\,\dot{\beta}\dot{\alpha}\sin(\alpha)\cos(\alpha) = M_\beta, \tag{4.54b}$$

where α, β are the previously described angular positions, $\dot{\alpha}$, $\dot{\beta}$ are their first time derivatives, $\ddot{\alpha}$, $\ddot{\beta}$ second time derivatives, ω is the angular speed of the disk's rotation, M_α, M_β are the motor torques, and the parameters involved in the model are given in Table 4.1. In what follows, the angular speed of the disk ω will be considered constant with value $\omega = 400\frac{2\pi}{60}\frac{rad}{s}$ which is maintained by the motor that controls the disc's rotating speed. The state vector is thus $x = [\alpha, \dot{\alpha}, \beta, \dot{\beta}]^T$ and the input vector is $M = [M_\alpha, M_\beta]^T$.

The controller will be synthesized in discrete-time for a sampling period $T_s = 20\,ms$. To formulate the optimization problem (4.35), it is necessary to be able to compute for each

generated scenario x its consecutive x^+ (the value of state after T_s). For each generated scenario x, its corresponding x^+ will be computed by using Matlab's ordinary differential equations ODE45 solver assuming constant input torques M_α, M_β over the period T_s.

Algorithm 2 Cascaded linear controller with saturatioin.

Require: state variables α , β , $\dot{\alpha}$, $\dot{\beta}$, matrices K^p , K^s ;

1: *Position controller*:

$$\begin{bmatrix} s_\alpha \\ s_\beta \end{bmatrix} = K^p \begin{bmatrix} \alpha \\ \beta \end{bmatrix};$$

2: *Speed controller*:

$$\begin{bmatrix} \tilde{M}_\alpha \\ \tilde{M}_\beta \end{bmatrix} = K^s \begin{bmatrix} s_\alpha - \dot{\alpha} \\ s_\beta - \dot{\beta} \end{bmatrix};$$

3: *Input saturation*:
Saturate $\tilde{M}_\alpha$ and $\tilde{M}_\beta$ by $M_{\text{sat}}(\gamma)$ in (4.55);

return $M_\alpha = M_{\text{sat}}(\tilde{M}_\alpha)$ and $M_\beta = M_{\text{sat}}(\tilde{M}_\beta)$.

The considered control policy corresponds to the block diagram given in Fig. 4.5 and is summarized in Algorithm 2. The input to the control policy is the state vector $x = [\alpha, \beta, \dot{\alpha}, \dot{\beta}]^T$ and the output is the vector of torques $M = [M_\alpha, M_\beta]^T$. To obtain M_α and M_β, the position loop first multiplies the position vector $[\alpha, \beta]^T$ by matrix $K^p \in \mathbb{R}^{2\times 2}$ whose components k^p_{ij}, $i, j \in \{1,2\}$ are the tuning parameters of the controller (i.e., they belong to the vector of tuning parameters η). The result of the multiplication is the speed reference vector $[s_\alpha, s_\beta]^T$. The difference of the speed references $[s_\alpha, s_\beta]^T$ and the speed values $[\dot{\alpha}, \dot{\beta}]^T$ is then multiplied by the matrix $K^s \in \mathbb{R}^{2\times 2}$ whose components k^s_{ij}, $i, j \in \{1,2\}$ are as well the tuning parameters of the controller (and are thus contained in η). The obtained vector of desired torques $[\tilde{M}_\alpha, \tilde{M}_\beta]^T$ is then passed through the input saturation block which saturates each of the two components by the function

$$M_{\text{sat}}(\gamma) = \begin{cases} -M_{\max}, & \text{if } \gamma \le -M_{\max} \\ \gamma, & \text{if } -M_{\max} \le \gamma \le M_{\max} \\ M_{\max}, & \text{if } M_{\max} \le \gamma \end{cases}, \tag{4.55}$$

where γ represents the component which is being saturated and $M_{\max}$ is the maximal applicable torque whose value is $M_{\max} = 0.2\,\text{Nm}$ for both of the motors. The output of the input-saturation block is the vector of torques $[M_\alpha, M_\beta]^T$ that is applied to the gyroscope. The controller's tuning parameters (the elements of vector η) are thus the elements of the K^p and K^s matrices, resulting in $\eta \in \mathbb{R}^8$.

The Lyapunov conditions (4.15) are selected to have the form:

$$V(x) - V(x^+) \geq \|x\|_2^2, \tag{4.56a}$$
$$V(x) \geq \|x\|_2^2, \tag{4.56b}$$

where the Lyapunov function is a function of the state vector x since the controller does not have state variables and thus it holds that $\theta = x$.

The set Ξ over which a Lyapunov function is searched for is specified to involve bounds on the angular positions $|\alpha| \leq \alpha_{\max}$, $|\beta| \leq \beta_{\max}$ with $\alpha_{\max} = \beta_{\max} = \pi/2$, as well as bounds on the angular speeds $|\dot{\alpha}| \leq \dot{\alpha}_{\max}$, $|\dot{\beta}| \leq \dot{\beta}_{\max}$ with $\dot{\alpha}_{\max} = \dot{\beta}_{\max} = \pi$. The local set Ξ is thus defined by $\Xi = \{x \,|\, \psi \geq 0\}$ where the function ψ is

$$\psi = \begin{bmatrix} \alpha + \alpha_{\max} \\ -\alpha + \alpha_{\max} \\ \beta + \beta_{\max} \\ -\beta + \beta_{\max} \\ \dot{\alpha} + \dot{\alpha}_{\max} \\ -\dot{\alpha} + \dot{\alpha}_{\max} \\ \dot{\beta} + \dot{\beta}_{\max} \\ -\dot{\beta} + \dot{\beta}_{\max} \end{bmatrix}. \tag{4.57}$$

The control synthesis is run with polynomial Lyapunov function $V(x)$ where the basis functions are all monomials of degree two. Such choice of the vector of basis functions $M(x)$ in (4.24) results in $c_{\mathrm{v}} \in \mathbb{R}^{10}$ (i.e., $n_{\mathrm{v}} = 10$). The search ranges for tuning parameters in η (i.e., for the coefficients of K^p and K^s matrices) are chosen to be $[-1, 1]$ for each of the tuning parameters, which was embedded into the set $\mathbf{D}$ of (4.41) and (4.39). The accuracy of the Lyapunov function computation in (4.35) is selected to $\Lambda_{\%} = 99.9\%$, which together with $\hat{\sigma} = 10^{-7}$ results in $N_{\mathrm{sc}} = 52237$ scenarios according to (4.38).

After 1000 Bayesian Optimization iterations applied to (4.41) as a part of phase one, 47 parameters η in the set $\{\eta \,|\, \mathrm{I}_p(\eta) = 0, \eta \in \mathbf{D}\}$ were obtained, with an average time per Bayesian optimization iteration of about 1.45 minutes (the computing platform involved 3.0GHz Intel Core i7 processor and 16GB of RAM). Note that the scenarios x for (4.35) and their corresponding pairs x^+ can be generated in a parallel manner which as such may decrease the required time per Bayesian iteration, but this possibility was not used in the examples of this chapter. The obtained parameters were then used as initial points in a total of 1500 Bayesian optimization iterations applied to the performance optimization problem (4.39), with the average time per iteration of about 2.3 minutes. The performance criteria $\mathrm{P}(\eta)$ used was a sum of the trajectory costs obtained from three initial states x_0 which were $0.6[\frac{\pi}{2}, 0, \frac{\pi}{2}, 0]^T$,

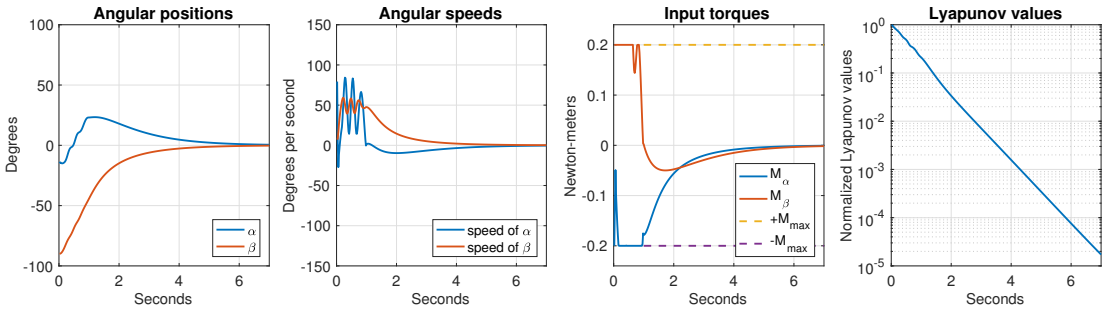


Figure 4.6 – Evolution of the gyroscope with the obtained controller starting from a randomly generated initial state x that involves $\alpha = -14.94$, $\beta = -89.98$, $\dot{\alpha} = 79.32$, $\dot{\beta} = 7.12$, where the values of positions α, β are given in degrees and the values of speeds $\dot{\alpha}$, $\dot{\beta}$ in degrees per second.

$0.6[\frac{\pi}{2},0,0,0]^T$ and $0.6[0,0,\frac{\pi}{2},0]^T$. Each trajectory cost had the form

$$\sum_{k=0}^{N_{\text{st}}} l_1(x_k) + l_2(x_k, x_{k-1}) + l_3(M_k, M_{k-1}), \tag{4.58}$$

where the number of simulation steps was $N_{\text{st}} = 500$ and the stage-cost terms were

$$l_1(x_k) = x_k^T Q_1 x_k, \tag{4.59a}$$
$$l_2(x_k, x_{k-1}) = (x_k - x_{k-1})^T Q_2 (x_k - x_{k-1}), \tag{4.59b}$$
$$l_3(M_k, M_{k-1}) = (M_k - M_{k-1})^T Q_3 (M_k - M_{k-1}), \tag{4.59c}$$

with diagonal matrix $Q_1 \in \mathbb{R}^{4\times4}$ whose diagonal elements were $(1,0,1,0)$ resulting thus in l_1 term which penalizes deviations of position angles from the target, diagonal matrix $Q_2 \in \mathbb{R}^{4\times4}$ whose diagonal elements were $(0,0.1,0,0.1)$ resulting thus in l_2 term which penalizes changes in angular speeds, and identity matrix $Q_3 \in \mathbb{R}^{2\times2}$ which penalizes the changes in the input signal.

The obtained controller was tested in simulation by applying it with the control period $T_s = 20\,\text{ms}$ and starting it from many random initial points. Fig. 4.6 shows the evolutions of the position angles (α, β), angular speeds $(\dot{\alpha}, \dot{\beta})$, input torques (M_α, M_β) and values of the calculated Lyapunov function $V(x)$ where the gyroscope was started from a randomly generated initial state. It can be seen that the used random initial state x_0 is outside of the range around the target point where the plant model would be approximatively linear, that the plant's input constraints are satisfied, and that the values of the Lyapunov function are decreasing along the trajectory.

The controller was implemented in an experimental setup by using LabVIEW for control of the Quanser gyroscope shown in Fig. 4.4. The angular positions were measured by encoders, and the angular speeds were calculated by using the difference of the last two measured positions

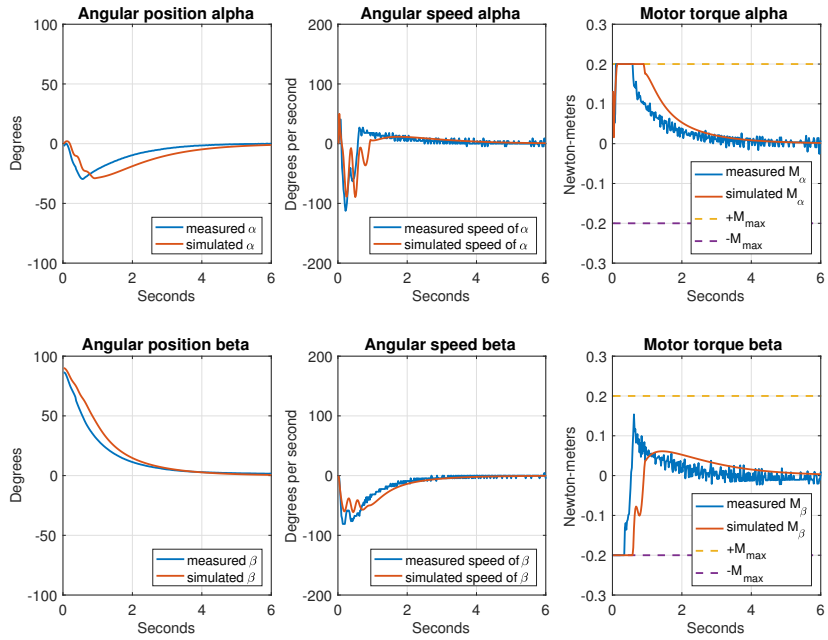


Figure 4.7 – Experimentally measured (blue) and simulated (red) results of the gyroscope in closed-loop with the obtained controller starting from initial state x that involves angular positions $\alpha = 0$, $\beta = 90$ and angular speeds $\dot{\alpha} = 0$, $\dot{\beta} = 0$ (the values given in degrees and degrees per second, respectively).

divided by the sampling time. The obtained experimental measurements are given in Fig. 4.7 together with simulated results. It can be seen that the input torques contain noise (originating from the noise present in the angular speeds fed back to the controller), which is however due to the noise high-frequency nature not expected to significantly influence the evolution of the states due to the low-pass filter nature of the mechanical system.

4.5 Conclusions

This chapter presented a method that extends the benefits of the automatic tuning procedure presented in Chapter 2 to problems of larger size. The obtained method also provides better modelling flexibility than the SOS approach as it does not require polynomial form for the functions describing the system. Instead of using the SOS programming technique for computation of Lyapunov functions, a procedure allowing a high accuracy numerical Lyapunov function estimation is introduced. The procedure formulates the Lyapunov function search as a robust optimization problem which is then tackled by applying a Chebychev center and scenario-based optimization techniques. The concept is then incorporated into a controller synthesis method whose demonstration is performed by application to two examples which cannot be addressed by the SOS-based approach due to scalability and modelling limitations.

Part II

Accelerated ADMM based on Accelerated Douglas-Rachford Splitting

5 Accelerated ADMM based on Accelerated Douglas-Rachford Splitting

5.1 Introduction

The alternating direction method of multipliers (ADMM) is an optimization algorithm for convex problems that has received an intensive attention in recent years due to its applicability to large-scale machine learning and image processing problems [16]. Even though developed a long time ago, the reasons for its renewed attention lie in its form conductive to distributed-memory implementation, its possibility of formulating closed form solutions for subproblems involved in the algorithm, and its practical performance which produces solutions with accuracy sufficiently high for many applications of interest, as summarized in [29].

One of the two ADMM optimization models that commonly appear in the literature is the Fenchel primal:

$$\begin{array}{ll} \text{minimize} & f_1(x) + f_2(Ax) \\ \text{subject to} & x \in \mathbb{R}^n, \end{array} \tag{5.1}$$

where $f_1 : \mathbb{R}^n \to (-\infty, \infty]$ and $f_2 : \mathbb{R}^m \to (-\infty, \infty]$ are closed proper convex functions, A is an $m \times n$ matrix, and a feasible solution x is assumed to exist. The augmented Lagrangian for (5.1) is

$$L_c(x, z, \lambda) = f_1(x) + f_2(z) + \langle \lambda, Ax - z \rangle + \frac{c}{2} \|Ax - z\|^2, \tag{5.2}$$

where $\lambda \in \mathbb{R}^m$ is a dual variable, $c > 0$ is a penalty parameter, and $\langle a, b \rangle$ denotes the inner product $a^T b$. The ADMM takes the form

$$x_{k+1} \in \underset{x \in \mathbb{R}^n}{\arg\min}\, L_c(x, z_k, \lambda_k), \tag{5.3a}$$

$$z_{k+1} \in \underset{z \in \mathbb{R}^m}{\arg\min}\, L_c(x_{k+1}, z, \lambda_k), \tag{5.3b}$$

$$\lambda_{k+1} = \lambda_k + c(Ax_{k+1} - z_{k+1}). \tag{5.3c}$$

The literature related to convergence of ADMM can be divided in two categories [13, 29]. The first approaches convergence analysis by exploiting only elementary mathematical principles and simple algebra to develop the conditions which lead to convergence of the algorithm [9, 16]. Even though it is based only on elementary principles, due to the intricate algebraic manipulations involved in the development, this approach is mostly regarded as being incapable of revealing the "deep structure" of the algorithm.

The second approach to analyse ADMM convergence is based on its relation to the generalised proximal point algorithm for finding a zero of a set-valued maximal monotone mapping [29, 30]. This relation is established by first observing that ADMM is a special case of Douglas-Rachford splitting, and then by showing that Douglas-Rachford splitting is a special case of the generalised proximal algorithm [30]. The established relation has allowed transfer of knowledge regarding the convergence of the generalised proximal point algorithm to the convergence of ADMM, which resulted in generalised ADMM involving inexact minimization of subproblems and the possibility for overrelaxation [30]. Moreover, it revealed that the basic engine behind ADMM is the same as that of the proximal point algorithm involved in the augmented Lagrangian method, and emphasised that viewing ADMM as an inexact minimization of the augmented Lagrangian involving only one cycle of block-coordinate descent is misleading [13, 29].

A general property of first order methods, including the gradient method and ADMM, is their slow convergence rate in comparison to advanced second order methods. A substantial contribution to the performance of a gradient method is achieved through its acceleration based on a sophisticated extrapolation rule between subsequent gradient steps. In the case of a gradient projection method which applies to minimization of a convex differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ over a closed convex set $X \subset \mathbb{R}^n$ with f having a Lipschitz continuous gradient satisfying

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \quad \forall\, x, y \in \mathbb{R}^n, \tag{5.4}$$

the version accelerated by extrapolation [13, 52] takes the form

$$y_k = x_k + \beta_k(x_k - x_{k-1}), \tag{5.5a}$$
$$x_{k+1} = P_X\big(y_k - \alpha\nabla f(y_k)\big), \tag{5.5b}$$

where $P_X : \mathbb{R}^n \to \mathbb{R}^n$ denotes projection on the set X, value $x_{-1} = x_0$, α is a suitably chosen stepsize and $\{\beta_k\}$ represents a sequence of extrapolation parameters. Under some very particular choices of the extrapolation $\{\beta_k\}$, [13, 52], one of which is defined as

$$\beta_0 = 0, \quad \beta_k = \frac{k-1}{k+2}, \quad \forall\, k \geq 1, \tag{5.6}$$

a $O(1/k^2)$ iteration complexity is attained, improving the $O(1/k)$ complexity of the non-accelerated gradient method which is obtained when $\{\beta_k\}$ is $\beta_k \equiv 0$ in (5.5). For some positive

constant q, the iteration complexity $O(1/k^q)$ is defined as an upper bound [13]:

$$\min_{l\leq k} f(x_l) \leq f^* + \frac{m}{k^q}, \tag{5.7}$$

in which $\{x_k\}$ is any sequence that the algorithm can produce and m is a positive constant that may depend on the problem data and the starting point x_0.

The acceleration technique of the gradient method has been successfully extended to acceleration of the alternating minimization algorithm (AMA), an algorithm with a form similar to ADMM, and this resulted in the fast alternating minimization algorithm (FAMA) [4, 37]. An upgrade of ADMM by an extrapolation technique has been analysed in [37], where it has been shown that if both functions involved in the cost are strongly convex with the second one being quadratic, the dual function values converge with $O(1/k^2)$ iteration complexity provided that the penalty parameter c is small enough.

This chapter derives an accelerated version of ADMM by using a recently proposed accelerated Douglas-Rachford (DR) splitting [57] on the Fenchel dual problem. The obtained method replaces the internal proximal point algorithm of classical ADMM by the accelerated gradient method applied on a specially constructed scaled DR envelope function [57]. The derived algorithm addresses the optimization model (5.1) with an assumption that the function $f_2(z)$ is a strongly convex quadratic, and involves a $O(1/k^2)$ bound on the values of the scaled DR envelope function when an upper bound on the penalty parameter is satisfied. A heuristic modification of the obtained method which can potentially extend the benefit of extrapolation for penalties beyond the upper bound is provided in the numerical results section.

The content of this chapter is based on the publication [58] and is structured as follows. Section 5.2 summarizes some theoretical results that will be used in the later analysis. Section 5.3 derives the accelerated ADMM algorithm based on accelerated DR splitting. Section 5.4 contains numerical experiments. Section 5.5 presents conclusions.

5.2 Theoretical Tools

5.2.1 Accelerated DR splitting

This section summarizes the results of [57] by expressing them in a notation which will be used in the later developments. The DR splitting addresses problems of the form

$$\begin{array}{ll} \text{minimize} & d(\lambda) = d_1(\lambda) + d_2(\lambda) \\ \text{subject to} & \lambda \in \mathbb{R}^m, \end{array} \tag{5.8}$$

where $d_1 : \mathbb{R}^m \to (-\infty, \infty]$ and $d_2 : \mathbb{R}^m \to (-\infty, \infty]$ are closed proper convex functions. Given $v_0 \in \mathbb{R}^m$, the DR splitting algorithm consists of the steps

$$\lambda_k = \text{prox}_{cd_2}(v_k), \tag{5.9a}$$

$$\mu_k = \text{prox}_{cd_1}(2\lambda_k - v_k), \tag{5.9b}$$

$$v_{k+1} = v_k + \rho_k(\mu_k - \lambda_k), \tag{5.9c}$$

where $c > 0$ is a penalty parameter, $\{\rho_k\} \subset [0,2]$ is a sequence of relaxation factors, and the expression

$$\text{prox}_{ch}(v) = \underset{z \in \mathbb{R}^m}{\arg\min} \left\{ h(z) + \frac{1}{2c}\|z - v\|^2 \right\} \tag{5.10}$$

defines the proximal operator $\text{prox}_{ch} : \mathbb{R}^m \to \mathbb{R}^m$ which is single valued and whose existence is guaranteed when the function $h : \mathbb{R}^m \to (-\infty, \infty]$ is closed proper convex. Assuming that the set of optimal solutions of (5.8) is nonempty, that $0 < \inf_k \{\rho_k\} \leq \sup_k \{\rho_k\} < 2$, and that the relative interiors of $\text{dom}(d_1)$ and $\text{dom}(d_2)$ have a point in common where

$$\text{dom}(d) = \{\lambda \mid d(\lambda) < \infty\} \tag{5.11}$$

represents the effective domain of a function d, the sequence $\{v_k\}$ converges to a fixed point of the DR splitting v^* which is related to $\lambda^* \in \arg\min\{d(\lambda)\}$ as $\lambda^* = \text{prox}_{cd_2}(v^*)$. Because of the relation between the fixed point of the DR splitting v^* and the optimal solution λ^*, finding a fixed point v^* is essentially the same as finding an optimal solution λ^* of the problem (5.8).

The following supplementary assumption ensures the existence of a convex differentiable function referred to as DR envelope, which plays the key role in development of the accelerated DR splitting.

Assumption 1. *The function $d_2(\lambda)$ and the penalty c satisfy*

$$d_2(\lambda) = \frac{1}{2}\lambda^T Q\lambda + q^T\lambda, \qquad c < \tfrac{1}{L_{d_2}}, \tag{5.12}$$

where $Q \in \mathbb{R}^{m\times m}$ is symmetric positive semidefinite, $q \in \mathbb{R}^m$, and L_{d_2} is the Lipschitz constant of the function d_2 (i.e., the maximal eigenvalue of Q).

The DR envelope $F_c^{DR}(v)$ whose existence is guaranteed under this additional assumption takes the form

$$F_c^{DR}(v) = d_2^c(v) - c\|\nabla d_2^c(v)\|^2 + d_1^c(v - 2c\nabla d_2^c(v)),$$

with $v \in \mathbb{R}^m$, and $h^c(v)$ denoting a Moreau envelope of a function $h(v)$ defined by

$$h^c(v) = \inf_{x \in \mathbb{R}^m} \left\{ h(x) + \frac{1}{2c}\|x - v\|^2 \right\}. \tag{5.13}$$

The set of stationary points of the DR envelope $F_c^{DR}(v)$ (i.e., of points $v^* \in \arg\min\{F_c^{DR}(v)\}$) coincides with the set of fixed points of the DR splitting.

The significance of the DR envelope is that one iteration of the scaled gradient method

$$v_{k+1} = v_k - \rho_k D \nabla F_c^{DR}(v_k) \tag{5.14}$$

is equivalent to one iteration of the DR splitting (5.9), provided that $D = c(2(I+cQ)^{-1} - I)^{-1}$ and the employed $\{\rho_k\}$ is the same. By introducing a scaled variable w defined by $v = Sw$ using $S = D^{\frac{1}{2}}$, the iteration (5.14) can be written as

$$w_{k+1} = w_k - \rho_k \nabla h(w_k) \tag{5.15}$$

where $h(w) = F_c^{DR}(Sw)$. Since it can be shown that the Lipschitz constant of $\nabla h(w)$ is

$$L_h = \frac{1 + cL_{d_2}}{1 - cL_{d_2}}, \tag{5.16}$$

it follows by convergence theory of the gradient method with constant stepsize [13] that the algorithm converges by choosing $\rho_k \in (0, 2/L_h)$.

The established equivalence of one DR splitting cycle with one iteration of the gradient method applied on $h(w)$ allows the introduction of accelerated DR splitting, which is obtained by applying the accelerated gradient method on the function $h(w)$. Given $y_0 = v_0 \in \mathbb{R}^m$, the resulting accelerated DR splitting algorithm takes the form

$$\lambda_k = \text{prox}_{cd_2}(y_k), \tag{5.17a}$$
$$\mu_k = \text{prox}_{cd_1}(2\lambda_k - y_k), \tag{5.17b}$$
$$v_{k+1} = y_k + \rho_k(\mu_k - \lambda_k), \tag{5.17c}$$
$$y_{k+1} = v_{k+1} + \beta_k(v_{k+1} - v_k). \tag{5.17d}$$

It can be shown that under the extrapolation rule $\{\beta_k\}$ given in (5.6), the accelerated version has a $O(1/k^2)$ iteration complexity guaranteeing that

$$d(\mu_k) - d^* \leq F_c^{DR}(v_k) - F_c^{DR*} \leq \frac{2}{c\rho(k+2)^2} \| v_0 - v^* \|^2, \tag{5.18}$$

where $F_c^{DR*} = F_c^{DR}(v^*) = d^* = \inf_{\lambda \in \mathbb{R}^m}\{d(\lambda)\}$ and $\rho := \rho_k \equiv 1/L_h$. By considering the strong convexity properties of $F_c^{DR}(v)$ [57] and utilising the extrapolation rule $\{\beta_k\}$ from [52], a linear convergence rate can as well be established. For the details related to development of the previous results, the reader is referred to [57].

5.2.2 Augmented Lagrangian and proximal algorithm

The proximal iteration

$$\lambda = \text{prox}_{cd}(\mu), \qquad \lambda, \mu \in \mathbb{R}^m, \tag{5.19}$$

where $d : \mathbb{R}^m \to (-\infty, \infty]$ is closed proper convex, can be shown to be equivalent to finding a decomposition [13] of the form

$$\mu = \lambda + cm, \qquad m \in \partial d(\lambda), \tag{5.20}$$

where $\partial d(\lambda)$ denotes the subdifferential of the function d at λ, which represents the set of all subgradients $g \in \mathbb{R}^m$ at λ satisfying $d(\gamma) \geq d(\lambda) + g^T(\gamma - \lambda)$, $\forall \gamma \in \mathbb{R}^m$.

In the case where the function $d(\lambda)$ is defined by

$$d(\lambda) = -\inf_{x \in \mathbb{R}^n} \left\{ h(x) + \langle \lambda, Ax - b \rangle \right\}, \tag{5.21}$$

where $h : \mathbb{R}^n \to (-\infty, \infty]$ is closed, proper and convex, the proximal iteration on the function $d(\lambda)$ can be evaluated in a way which involves minimization of the augmented Lagrangian function, as described in, for example, [29]:

Proposition 1. *(*[29], Proposition 9*) Given any $\mu \in \mathbb{R}^m$, consider the problem*

$$\inf_{x \in \mathbb{R}^n} \left\{ h(x) + \langle \mu, Ax - b \rangle + \frac{c}{2} \| Ax - b \|^2 \right\}. \tag{5.22}$$

If $\bar{x}$ is an optimal solution to this problem, then setting $\lambda = \mu + c(Ax - b)$ and $m = b - A\bar{x}$ yields $\lambda, m \in \mathbb{R}^m$ such that $\mu = \lambda + cm$ and $m \in \partial d(\lambda)$, where $d(\lambda)$ is as defined in (5.21).

5.3 Accelerated ADMM based on Accelerated DR Splitting

The derivation of the accelerated ADMM algorithm based on the accelerated DR splitting (5.17) from [57] will be performed in a way inspired by the development of standard ADMM in [29]. The outcome will be a modified ADMM algorithm whose underlying working mechanism is characterised by the $O(1/k^2)$ iteration complexity given in (5.18). The optimization model addressed by the method to be developed is the Fenchel primal form given in (5.1), which can be reformulated as

$$\begin{array}{ll} \text{minimize} & f_1(x) + f_2(z) \\ \text{subject to} & Ax = z. \end{array} \tag{5.23}$$

By applying the standard Lagrange duality for equality constrained problems [12], one obtains the Lagrange dual function:

$$\begin{aligned} q(\lambda) &= \inf_{x\in\mathbb{R}^n, z\in\mathbb{R}^m} \{f_1(x) + f_2(z) + \lambda^T(Ax - z)\} \\ &= \inf_{x\in\mathbb{R}^n} \{f_1(x) + (A^T\lambda)^T x\} + \inf_{z\in\mathbb{R}^m} \{f_2(z) - \lambda^T z\} \\ &= -\sup_{x\in\mathbb{R}^n} \{(-A^T\lambda)^T x - f_1(x)\} - \sup_{z\in\mathbb{R}^m} \{\lambda^T z - f_2(z)\} \\ &= -f_1^\star(-A^T\lambda) - f_2^\star(\lambda) = -d_1(\lambda) - d_2(\lambda), \end{aligned}$$

in which $d_1(\lambda) = f_1^\star(-A^T\lambda)$, $d_2(\lambda) = f_2^\star(\lambda)$, and the $f^\star(\lambda) = \sup_{z\in\mathbb{R}^m} \{\lambda^T z - f(z)\}$ represents the conjugate of a function f. This leads to a Fenchel dual problem

$$\begin{array}{ll} \text{minimize} & d(\lambda) = d_1(\lambda) + d_2(\lambda) \\ \text{subject to} & \lambda \in \mathbb{R}^m, \end{array} \tag{5.24}$$

which minimizes $d(\lambda) = -q(\lambda)$ (i.e., maximises the dual function $q(\lambda)$ defined above).

Accelerated ADMM is derived by applying the accelerated DR splitting (5.17) to the Fenchel dual problem (5.24). For this purpose, (5.24) should satisfy the assumptions of Section 5.2.1 guaranteeing the existence of the DR envelope $F_c^{DR}(v)$.

Proposition 2. *The functions $d_1(\lambda)$ and $d_2(\lambda)$ are closed, convex and proper. Moreover, under the assumption that the function $f_2(z)$ is a strongly convex quadratic and the penalty c is small enough:*

$$f_2(z) = \frac{1}{2} z^T P z + p^T z, \qquad c < \frac{1}{L_{d_2}}, \tag{5.25}$$

where $P \in \mathbb{R}^{m\times m}$ is symmetric positive definite, $p \in \mathbb{R}^m$ and where L_{d_2} is a Lipschitz constant of the function d_2, the conditions of the Assumption 1 are satisfied, ensuring the existence of the DR envelope $F_c^{DR}(v)$.

Proof. The closedness and convexity of $d_1(\lambda)$ and $d_2(\lambda)$ follow from Lemma 8 in [29], and their properness from Prop. 1.6.1(b) in [12]. Since by the expression for conjugate of a strongly convex quadratic [73] there holds

$$d_2(\lambda) = f_2^\star(\lambda) = \frac{1}{2}\lambda^T P^{-1}\lambda - p^T P^{-1}\lambda, \tag{5.26}$$

the function $d_2(\lambda)$ is strongly convex quadratic as P^{-1} of a positive definite matrix P is positive definite. Considering as well the bound imposed on the penalty c, the conditions of Assumption 1 are satisfied. □

The following lemma derives a form of the accelerated ADMM based on the accelerated DR splitting. For this purpose, an additional assumption is introduced providing two sufficient

conditions that ensure the existence of the optimal solution in the minimization (5.27a) below.

Assumption 2. *It holds that the effective domain* $\mathrm{dom}(f_1) = \{x \mid f_1(x) < \infty\}$ *is bounded and/or that* $A^T A$ *is invertible.*

Lemma 1. *Given* $z_0, \lambda_0 \in \mathbb{R}^m$, $c > 0$, *a sequence* $\{\rho_k\} \subset [0,2]$ *and a sequence* $\{\beta_k\}$, *the sequence of recursions applied to the Fenchel primal* (5.1)*:*

$$x_{k+1} \in \underset{x\in\mathbb{R}^n}{\arg\min} \left\{ f_1(x) + \langle \lambda_k, Ax\rangle + \frac{c}{2}\|Ax - z_k\|^2 \right\}, \tag{5.27a}$$

$$z_{k+1} \in \underset{z\in\mathbb{R}^m}{\arg\min} \left\{ f_2(z) - \langle \lambda_k + E_k, z\rangle + \frac{c}{2}\left\| \frac{1}{c}\xi_k - z \right\|^2 \right\}, \tag{5.27b}$$

$$\lambda_{k+1} = \lambda_k + E_k + \xi_k - cz_{k+1}, \tag{5.27c}$$

with ξ_k, E_k *calculated as*

$$\xi_k = c(\rho_k A x_{k+1} + (1-\rho_k) z_k), \tag{5.28}$$

$$E_k = \beta_k(\lambda_k - \lambda_{k-1}) + \beta_k cA(\rho_k x_{k+1} - \rho_{k-1} x_k) + \beta_k c((1-\rho_k) z_k - (1-\rho_{k-1}) z_{k-1}), \tag{5.29}$$

where $E_0 := 0$, *is mathematically equivalent to the accelerated DR splitting* (5.17) *applied to the Fenchel dual* (5.24)*, and therefore equivalent to the accelerated gradient method applied to* $h(w) = F_c^{DR}(Sw)$.

Proof. The starting point for the development is the accelerated DR splitting (5.17). Given the initial $\lambda_0 \in \mathbb{R}^m$, $y_{-1} := \lambda_0 + cz_0$ and using $v_{-1} := v_0$ in the first iteration, (5.17) can be rewritten as

$$\mu_k = \mathrm{prox}_{cd_1}(2\lambda_k - y_{k-1}), \tag{5.30a}$$

$$v_k = y_{k-1} + \rho_k(\mu_k - \lambda_k), \tag{5.30b}$$

$$y_k = v_k + \beta_k(v_k - v_{k-1}), \tag{5.30c}$$

$$\lambda_{k+1} = \mathrm{prox}_{cd_2}(y_k), \tag{5.30d}$$

which in comparison to (5.17) has the λ_k update moved to the end, and the indices of v and y variable shifted from $k+1$ to k. By the equivalence of (5.19) and (5.20), the sequence (5.30) can be equivalently written as

$$\mu_k + cw_k = \lambda_k - cm_k, \qquad w_k \in \partial d_1(\mu_k), \tag{5.31a}$$

$$v_k = y_{k-1} + \rho_k(\mu_k - \lambda_k), \tag{5.31b}$$

$$y_k = v_k + \beta_k(v_k - v_{k-1}), \tag{5.31c}$$

$$\lambda_{k+1} + cm_{k+1} = y_k, \qquad m_{k+1} \in \partial d_2(\lambda_{k+1}), \tag{5.31d}$$

where $m_0 = z_0$, and $\lambda_k - cm_k$ in (5.31a) is obtained by substituting the y_k from (5.31d) for y_{k-1} in (5.30a). By applying the evaluation of proximal iterate by augmented Lagrangian from

Proposition 1, (5.31a) can be written as

$$x_{k+1} \in \underset{x\in\mathbb{R}^n}{\arg\min} \left\{ f_1(x) + \langle \lambda_k - cm_k, Ax\rangle + \frac{c}{2}\|Ax\|^2 \right\}, \tag{5.32a}$$

$$\mu_k = \lambda_k - cm_k + cAx_{k+1}, \tag{5.32b}$$

$$w_k = -Ax_{k+1}, \tag{5.32c}$$

as well as (5.31d) which becomes

$$z_{k+1} \in \underset{z\in\mathbb{R}^m}{\arg\min} \left\{ f_2(z) + \langle y_k, -z\rangle + \frac{c}{2}\|-z\|^2 \right\}, \tag{5.33a}$$

$$\lambda_{k+1} = y_k - cz_{k+1}, \tag{5.33b}$$

$$m_{k+1} = z_{k+1}. \tag{5.33c}$$

By Weierstrass theorem (Prop. 3.2.1, [12]), the existence of the solution of (5.32a) is ensured by the Assumption 2, and in case of (5.33a) the existence is ensured by the presence of the quadratic term $\|z\|^2$.

By using (5.33b) to express y_{k-1} and (5.32b), (5.33c) to express $\mu_k - \lambda_k$, (5.31b) can be written as

$$v_k = \lambda_k + cz_k + \rho_k c\,(Ax_{k+1} - z_k) = \lambda_k + c(\rho_k Ax_{k+1} + (1-\rho_k)z_k). \tag{5.34}$$

By introducing (5.34) into (5.31c) for v_k and v_{k-1}, one obtains

$$y_k = \lambda_k + \xi_k + E_k, \tag{5.35}$$

where ξ_k, E_k are defined in (5.28) and (5.29). By specifying $E_0 := 0$, the compliance of (5.35) with $v_{-1} := v_0$ assumed for (5.30) is obtained.

By using the expressions (5.32a)-(5.35), the cycle (5.31) gets expressed as

$$x_{k+1} \in \underset{x\in\mathbb{R}^n}{\arg\min} \{ f_1(x) + \langle \lambda_k - cz_k, Ax\rangle + \frac{c}{2}\|Ax\|^2 \}, \tag{5.36a}$$

$$y_k = \lambda_k + \xi_k + E_k, \tag{5.36b}$$

$$z_{k+1} \in \underset{z\in\mathbb{R}^m}{\arg\min} \{ f_2(z) + \langle y_k, -z\rangle + \frac{c}{2}\|-z\|^2 \}, \tag{5.36c}$$

$$\lambda_{k+1} = y_k - cz_{k+1}, \tag{5.36d}$$

with ξ_k, E_k defined as in (5.28), (5.29). By substituting (5.36b) into (5.36c) and (5.36d), and by adding the constant terms $\frac{c}{2}\|z_k\|^2$ and $\frac{c}{2}\|\frac{1}{c}\xi_k\|^2$ to (5.36a) and (5.36c), respectively, the equations (5.36) take the form (5.27). □

It can be noticed that by setting the extrapolation term $\beta_k \equiv 0$, the accelerated ADMM (5.27) reduces to the generalized ADMM [30] with relaxations, and by choosing as well the relaxations

$\rho_k \equiv 1$ one obtains the classical ADMM given in (5.3). The following proposition summarizes the preceding development, and provides a proof which is an adapted version of Proposition 15 in [29].

Proposition 3. *Consider the optimization model* (5.1) *in its equivalent form* (5.23). *Let Assumption 2 hold, and let the function $f_2(z)$ be strongly convex quadratic $f_2(z) = \frac{1}{2} z^T P z + p^T z$ with $P \in \mathbb{R}^{m\times m}$ symmetric positive definite and $p \in \mathbb{R}^m$. Assume that all subgradients of the function $d_1(\lambda) = -\inf_{x\in\mathbb{R}^n} \{f_1(x) + (A^T\lambda)^T x\}$ at each point $\lambda \in \mathbb{R}^m$ take the form $-A\bar{x}$ where $\bar{x}$ attains the stated minimum over x. Then, there exists a primal-dual optimal solution pair $((x^*, z^*), \lambda^*)$, and if the sequences $\{x_k\} \subset \mathbb{R}^n$, $\{z_k\} \subset \mathbb{R}^m$ and $\{\lambda_k\} \subset \mathbb{R}^m$ conform to the recursion* (5.27) *under the assumptions of Lemma 1 using* (5.6) *for $\{\beta_k\}$ and*

$$\rho_k \equiv \rho = \frac{1 - cL_{d_2}}{1 + cL_{d_2}}, \qquad c < \frac{1}{L_{d_2}}, \tag{5.37}$$

where L_{d_2} is the maximal eigenvalue of P^{-1}, then $\lambda_k \to \lambda^\infty$, $z_k \to z^\infty$ and $Ax_k \to Ax^\infty = z^\infty$ where x^∞ is a limit point of $\{x_k\}$ and $((x^\infty, z^\infty), \lambda^\infty)$ corresponds to a primal-dual solution pair $((x^*, z^*), \lambda^*)$. The algorithm is characterized by the $O(1/k^2)$ iteration complexity of the form (5.18).

Proof. The existence of the primal-dual optimal solution pair $((x^*, z^*), \lambda^*)$ follows from Prop. 1.2.1(a)-(b) in [13], where the relative interior conditions are satisfied due to the quadratic form of $f_2(z)$ and $d_2(\lambda)$.

The previous development shows that the recursion (5.27) is equivalent to the sequence $y_{k-1} = \lambda_k + cm_k = \lambda_k + cz_k$ from (5.33b)-(5.33c) produced by the accelerated DR splitting (5.30), which is furthermore equivalent to the application of the fast gradient method to the scaled DR envelope $h(w)$. Since the relaxation factor ρ is by (5.37) chosen in accordance with the Lipschitz constant of $\nabla h(w)$ given in (5.16), the sequence $\{v_k\}$ of (5.30) converges to the DR fixed point v^∞ and by (5.30c) the sequence $\{y_k\} \to v^\infty$, if the point v^∞ exists. Since the primal-dual optimal solution pair exists, the point $\lambda^* + cz^*$ is just such a point v^∞, so it exists. Therefore the sequence $\{y_k\}$ converges to a DR splitting fixed point $v^\infty = y^\infty$, and by Lemma 14 in [29] it has the form $y^\infty = \lambda^\infty + cz^\infty$ where $z^\infty \in \partial d_2(\lambda^\infty)$ and $-z^\infty \in \partial d_1(\lambda^\infty)$. By the assumption regarding the subgradients of d_1, there exists some x^∞ such that $-Ax^\infty = -z^\infty$, or equivalently $Ax^\infty = z^\infty$.

Since the proximal mapping $\text{prox}_{cd_2}(y)$ is nonexpansive (Prop. 5.1.8 in [13]), it is also continuous. We have $\text{prox}_{cd_2}(y^\infty) = \lambda^\infty$ and $\text{prox}_{cd_2}(y_k) = \lambda_k$, and because of continuity of $\text{prox}_{cd_2}(y)$ we also have $\lambda_k = \text{prox}_{cd_2}(y_k) \to \text{prox}_{cd_2}(y^\infty) = \lambda^\infty$ and threfore $z_k = (y_{k-1} - \lambda_k)/c \to z^\infty = (y^\infty - \lambda^\infty)/c$. By using the first equation in (5.34) together with $y_{k-1} = \lambda_k + cz_k \to v^\infty$, since $\rho > 0$ due to (5.37) there holds $Ax_{k+1} - z_k \to 0$, from where it follows that $Ax_{k+1} \to Ax^\infty = z^\infty$ with x^∞ being a limit point of $\{x_k\}$. □

5.4 Numerical Experiments

The algorithm is tested by solving a quadratic programming (QP) problem of the form

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T Px + p^T x \\ \text{subject to} & l_b \leq x \leq u_b, \end{array} \tag{5.38}$$

where $P \in \mathbb{R}^{n\times n}$ is symmetric positive definite, $p \in \mathbb{R}^n$, and the $l_b, u_b \in \mathbb{R}^n$ represent the lower and upper bound of the variable x, respectively. The QP data with $n = 100$ is generated randomly using Matlab commands; in particular, the components of p, l_b and u_b with $l_b \leq u_b$ are generated using normal distribution and P is generated by using $P = VMV^T$ where V contains the eigenvectors of a symmetric matrix whose coefficients are uniformly distributed and M is a diagonal matrix with elements equally spaced between 1 and 100, resulting thus in the matrix P that has the eigenvalues equal to the diagonal elements of M.

The QP (5.38) is expressed in the form (5.23) by setting $f_1(x) = \delta_X(x)$, $f_2(z) = \frac{1}{2}z^T Pz + p^T z$, where $\delta_X(x)$ is the indicator function [12] of the box constraint $X = \{x \mid l_b \leq x \leq u_b\}$. The equality constraint of (5.23) is thus $x - z = 0$.

The stopping criteria are derived as in Section 3.3 of [16]. The obtained primal and dual residual are

$$r_{k+1} = Ax_{k+1} - z_{k+1}, \tag{5.39a}$$

$$s_{k+1} = c(z_k - z_{k+1}) + E_k - c(1-\rho_k)(Ax_{k+1} - z_k). \tag{5.39b}$$

The stopping criteria used in the experiments are $\|r_k\| \leq \varepsilon^{pri}$ and $\|s_k\| \leq \varepsilon^{dual}$, with ε^{pri} and ε^{dual} chosen using the absolute and relative criterion from Section 3.3.1 of [16] with $\varepsilon^{abs} = 10^{-4}$, $\varepsilon^{rel} = 10^{-2}$. The maximal number of iterations for which the experiments are run is $k_{\max} = 10000$. The initial conditions are set to zero vectors of appropriate dimensions.

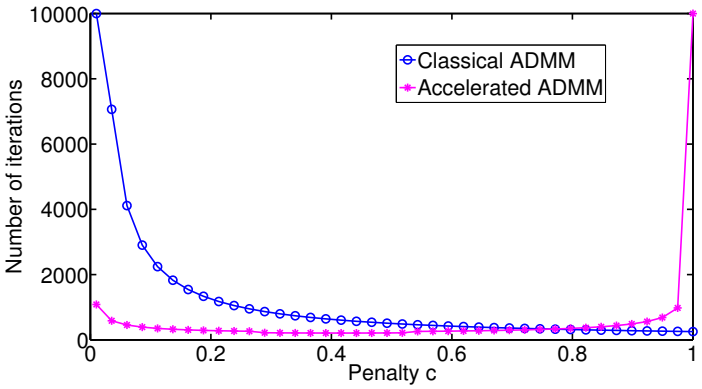
The comparison of classical ADMM (5.3) with the accelerated version over the range of penalties $c \in (0,1]$ which ensures convergence of the accelerated version by Prop. 3 is given in Fig. 5.1a and Fig. 5.2a. As the value of the penalty c approaches the upper bound $c_{\max} = 1/L_{d_2} = 1$ from (5.37), the relaxation ρ tends to 0 according to (5.37) and thus the accelerated version, characterised by the $O(1/k^2)$ complexity in (5.18), gradually worsens performance and eventually stops converging.

For this reason, a heuristic version which uses $\rho = 1$ for every value of c is introduced and tested. The results are given in Fig. 5.1b and Fig. 5.2b, where the heuristically accelerated ADMM (i.e., the method with $\rho = 1$) is compared with the classical ADMM (5.3), the generalised ADMM [29] with overrelaxation set to $\rho = 1.9$, and the Fast ADMM with and without restarting ([37], Algorithms 7 and 8). Nevertheless, the heuristic modification $\rho = 1$ may as well cause divergence, as can be seen on Fig. 5.1c and Fig. 5.2c where a QP with a random matrix P containing the eigenvalues equally spaced between 1 and 500 is considered. These results

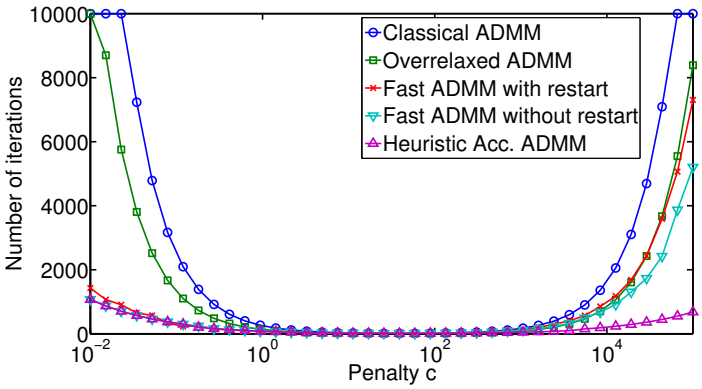
indicate that the heuristic version could benefit by introducing a restarting scheme, like the one of the Fast ADMM with restarting [37], which would ensure convergence while keeping the accelerated behaviour.

5.5 Conclusions

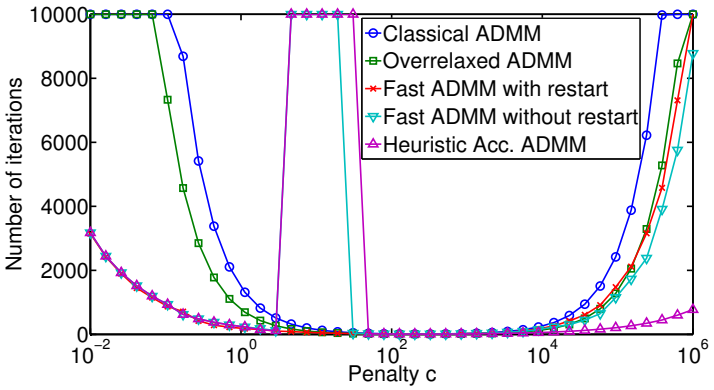
An accelerated version of ADMM based on accelerated Douglas-Rachford splitting is derived resulting in a method characterised by a $O(1/k^2)$ complexity of the internal convergence mechanism. In comparison to the classical ADMM, the derived algorithm involves an additional algebraic step which corresponds to the extrapolations in the underlying fast gradient method. The numerical results show that the method can improve the performance of classical ADMM over the allowed range of penalty parameters, and that a heuristic modification can potentially extend the benefits of acceleration beyond this allowed range.



(a) Classical and accelerated ADMM over the theoretically allowed range of penalties c. The condition number of QP is equal to 100.

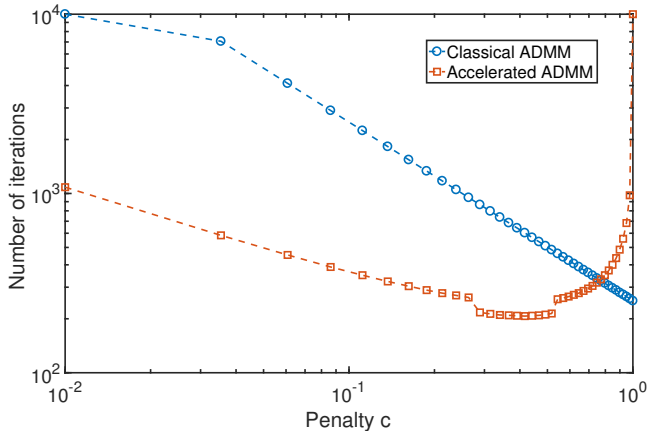


(b) Heuristically accelerated ADMM compared with other versions of ADMM. The same random QP as in (a) is considered.

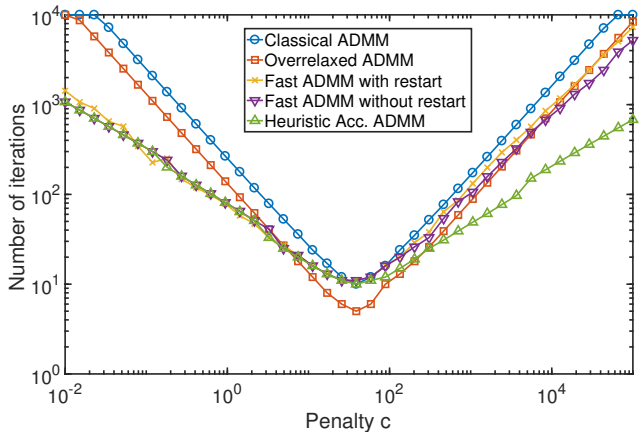


(c) The same methods as in (b), but applied on a QP whose condition number is 500. Divergence over a range of c can be observed.

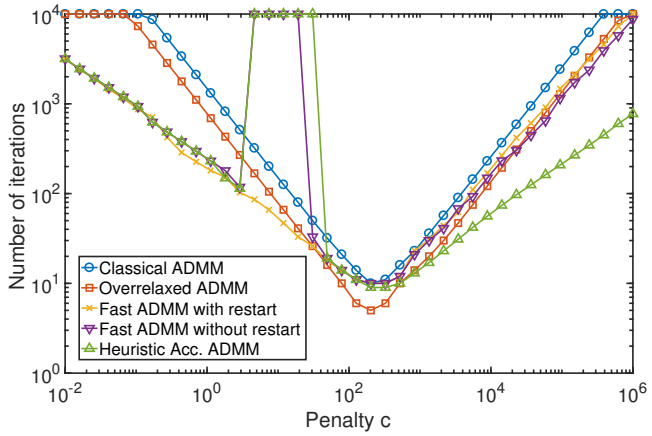
Figure 5.1 – Accelerated ADMM and its heuristic modification, applied for solving a random QP of the form (5.38).



(a) Classical and accelerated ADMM over the theoretically allowed range of penalties c. The condition number of QP is equal to 100.



(b) Heuristically accelerated ADMM compared with other versions of ADMM. The same random QP as in (a) is considered.



(c) The same methods as in (b), but applied on a QP whose condition number is 500. Divergence over a range of c can be observed.

Figure 5.2 – The same results as in Fig. 6.8 but with logarithmic axes.

Part III

Power Converter Pulse Pattern Optimal Control

6 Power Converter Pulse Pattern Optimal Control

6.1 Introduction

The field of medium-voltage power electronics has demonstrated itself as a fruitful ground for application of Model Predictive Control (MPC) [68]. In comparison to the traditional control approaches that exist in the field, the developed MPC schemes have shown an ability to get the existing power electronics hardware closer to its full operating potential, providing better efficiency and dynamic/steady-state performance [34]. To address the switching nature of power electronic systems and obtain controllers that are computationally tractable in real-time, the aforementioned MPC schemes consist of problem-tailored adaptations of the general MPC concept and involve many engineering approximations and simplifications.

One of the most prominent examples of medium-voltage power electronics MPC is Model Predictive Direct Torque Control (MPDTC) [35]. MPDTC is an extension of the classical Direct Torque Control (DTC) [71] and involves a replacement of the DTC's look-up table with an optimization problem that minimizes the converter's switching frequency along the prediction horizon. The method was experimentally tested on an industrial medium-voltage drive system in [55] and has experienced subsequent further refinements such as direct minimization of the converter's switching losses and more efficient solving of the involved optimization problem by using branch-and-bound technique [31]. Another prominent MPC scheme, which could be interpreted as an adaptation of the core MPDTC concept to the current control problem, is Model Predictive Direct Current Control (MPDCC) [33] which instead of the torque control provides control of the current and therefore can be used as an inner loop in the case of Field Oriented Control (FOC) [14, 38]. To overcome a weakness of MPDTC and MPDCC which both require computationally challenging long prediction horizons in order to achieve superior total harmonic distortion (THD) in steady-state operation, another prominent MPC scheme arose under the name Model Predictive Pulse Pattern Control (MP^3C) [36]

The MP^3C control scheme is conceptually closest to the method which will be the presented in this chapter. MP^3C was introduced in [36] and experimentally verified in [54] on a setup involving a five-level active neutral-point-clamped (ANPC) inverter. One of the principal

characteristics of MP^{3}C is its usage of offline computed optimal steady-state input signals that are also known in power electronics community as Optimized Pulse Patterns (OPPs) [18, 50, 67]. OPPs are generally considered as a mean whose utilization provides the best possible steady-state performance, with an ability to reduce the steady-state current THD even up to 50% in comparison to the traditional Space Vector Modulation (SVM) [32]. The usage of OPPs allows MP^{3}C to surpass the weakness of MPDTC and MPDCC, which is their need for long and computationally demanding prediction horizons for achieving superior steady-state performance. In the case of MP^{3}C, the optimal steady-state OPP input is used as a baseline whose switching times are reoptimized over a certain prediction horizon in order to achieve zero tracking error at the end of the prediction horizon. While long horizons of MP^{3}C turned out to provide better noise resistance in comparison to a deadbeat MP^{3}C version, they also result in slower dynamic performance due to the penalization of the tracking error only at the end of the prediction horizon and not along it. The internal model of MP^{3}C represents the power electronics system with two integrators, one for each of the two axes of the stationary coordinate frame.

The method of this chapter belongs to the category of power electronics MPC schemes where modulator (using SVM [23] or carrier-based PWM [40]) is eliminated, which is a category that also involves MPDTC, MPDCC and MP^{3}C. In comparison to the methods which use OPPs to store the optimal steady-state operation in controller's memory, the method of this chapter also allows a usage of memory to store the information about dynamic system behavior, thereby allowing an approximate low-computational complexity MPC that optimizes the transient behavior of complex power electronics configurations for which it is not possible to approximate the plant dynamics with two integrators, as is done in MP^{3}C. In comparison to MP^{3}C, the method is thus more general (e.g., directly applicable in presence of *LC* filters) and furthermore, it also penalizes the tracking error not only at the end of the prediction horizon but along it, thus not experiencing a degradation of transient performance when longer prediction horizons are used.

Besides its applicability to a large variation of different power electronics configurations, in order to facilitate an easier exposition the method is described in this chapter with a focus on an industrial power electronics case study involving a grid-tied converter with *LC* filter. It is shown that the method can control such a system without a need for an additional active damping loop [26, 27] by whose utilization the resonant behavior introduced with *LC* filter is often tackled (for an adaptation of the active damping concept to the case of MP^{3}C, see [39]), removing thereby a need for having the converter's switching frequency considerably larger than the resonant frequency of the system. The combinaiton of OPPs and *LC* filter allows achievement of very low THD values in steady-state, and as indicated in [34], with design of OPPs one can also minimize the size and cost of the involved *LC* filter.

The description of the method is done by first introducing an optimal control problem (OCP) which uses the OPPs as steady-state baseline signals, and the OCP is then tackled by using a sequential approach to optimal control and a gradient projection algorithm. To achieve

a low-computational approximation of the OCP, additional elements are then introduced. In particular, a possibility of memory storage of the dynamic system behavior necessary for efficient formulation of low-complexity OCP approximations is introduced, as well as a computationally efficient way to approximately solve the obtained optimization problem. The introduced approximations open a possibility for an implementation of some of the controller's versions on a field-programmable gate array (FPGA), for which some indications regarding the required computational resources are provided in the numerical results section of this chapter. The obtained approximate controller versions are tested in simulation by comparison with the unapproximated OCP based on OPPs, giving an insight into the influence which the introduced controller approximations have on the performance. The introduced concept of memory storage of the dynamic information can also be beneficially applied in case of a related method using OPPs in [3], which instead of an integral of the tracking error considers the tracking error at a finite number of points along the prediction horizon for minimization.

The work of this chapter is largly based on work published in [59] which was developed in collaboration with Stefan Almer and Helfried Peyrl from ABB. This chapter involves further refinement of the description from [59] as well as additional developments which allow more efficient approximate solving of the OCP, the description of which is primarily located in the subsections of Section 6.5. In collaboration with my colleague Harsh Shukla, one of the approximate versions of the method has been implemented on an FPGA by him, the description of which is left out of this thesis with only basic characteristics of the implementation outlined in Section 6.6.

The rest of the chapter is structured as follows. Section 6.2 describes the industrial power electronics case study considered in the chapter. Section 6.3 states assumptions and provides a mathematical model of the system. Section 6.4 describes basic notions and terminology of OPPs, introduces an OCP based on OPPs, applies sequential approach to optimal control to obtain an OCP form with a finite number of decision variables and describes application of gradient algorithm to the obtained optimization problem. Section 6.5 introduces concepts for achieving a low-computational approximation of the OCP, introducing among others memory storage of the dynamic information and computationally efficient approximate solving of the obtained optimization problem. Section 6.6 provides numerical experiments.

6.1.1 Notation

For given positive integers n and m, $\mathrm{I}_{n\times n} \in \mathbb{R}^{n\times n}$ and $0_{n\times m} \in \mathbb{R}^{n\times m}$ denote the identity and zero matrix of specified dimensions, respectively. For given vectors $a_i \in \mathbb{R}^{n_i}$, $i = 1,...,m$, the vector $c = [a_1^T,...,a_m^T]^T$ is denoted by $c = (a_1,...,a_m)$, which is thus an element of $\mathbb{R}^{n_1+...+n_m}$. For a three phase quantity $x_{abc} \in \mathbb{R}^3$ with components $x_{abc} = (x_a, x_b, x_c)$, the direct and inverse Clarke transform (with neglected zero sequence component) to the $\alpha\beta$ representation $x_{\alpha\beta} \in \mathbb{R}^2$

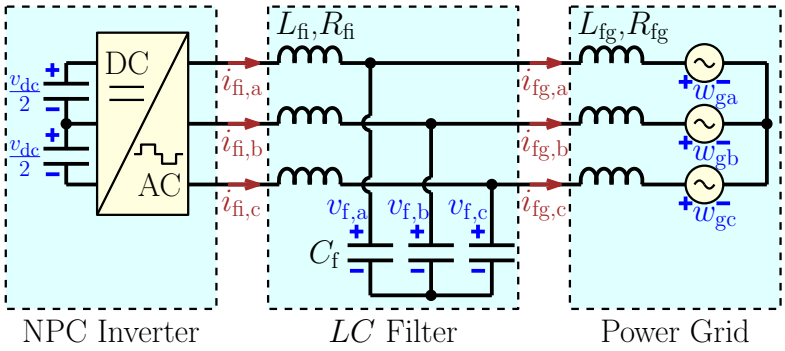


Figure 6.1 – Three-level neutral-point-clamped (NPC) voltage source inverter connected to the power grid via an LC filter.

with $x_{\alpha\beta} = (x_\alpha, x_\beta)$ are defined respectively by the matrices

$$P_{2\times 3} := \frac{2}{3}\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix}, \qquad P_{3\times 2}^{-1} := \begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix}, \tag{6.1}$$

where $x_{abc} = P_{2\times 3} x_{\alpha\beta}$ is direct and $x_{\alpha\beta} = P_{2\times 3}^{-1} x_{abc}$ inverse transform [25].

6.2 Grid-tied Converter with LC Filter Case Study

The control method will be presented by considering an industrial medium-voltage power electronics case study involving a grid-tied converter with LC filter. The configuration is illustrated in Fig. 6.1. The actuator in the system is the power converter (NPC inverter), and the goal of the controller is to ensure that a desired active and reactive power pair (P_g, Q_g) is injected into the grid.

The power converter is a three-phase three-level neutral-point-clamped (NPC) voltage source inverter, which is a diode clamped converter topology capable of producing three voltage levels per phase [51]. The NPC converter is interfaced to the power grid through an intermediate LC filter, which is an additional passive component whose insertion is often required in order to meet the relevant grid standards by reducing the harmonic distortions at the grid side. It should be noted that the involved LC filter does not include resistive elements for damping of resonance, which should therefore be tackled by the controller.

6.3 Mathematical Model of the System

6.3.1 System assumptions

The case study system is addressed by involving a set of assumptions that is common in the related literature. The assumptions concern the NPC inverter, *LC* filter and power grid. The NPC inverter is assumed to be supplied by a dc-link voltage whose total value is constant. The NPC inverter's fluctuations of the neutral point potential are neglected, as well as the voltage drops on the power semiconductor devices during conduction. The nonlinearity of the magnetic materials in the inductive elements will be neglected, as well as the temperature-dependence of the resistances. The grid voltage is assumed to be three-phase symmetric, and the grid inductance is assumed constant.

6.3.2 State-space model of the *LCL* circuit and grid voltage

The state of the *LCL* circuit in Fig. 6.1 is described in stationary *abc* frame with state vector

$$x_{\mathrm{f}} = (\, i_{\mathrm{fi},a}\,,\, i_{\mathrm{fi},b}\,,\, i_{\mathrm{fi},c}\,,\, i_{\mathrm{fg},a}\,,\, i_{\mathrm{fg},b}\,,\, i_{\mathrm{fg},c}\,,\, v_{\mathrm{f},a}\,,\, v_{\mathrm{f},b}\,,\, v_{\mathrm{f},c}\,) \tag{6.2}$$

where for each phase $p \in \{a,b,c\}$, the $i_{\mathrm{fi},p}$, $i_{\mathrm{fg},p}$ and $v_{\mathrm{f},p}$ respectively denote the inverter current, grid current and capacitor voltage, as illustrated in Fig. 6.1. The *LCL* circuit dynamics are

$$\dot{x}_{\mathrm{f}}(t) = A_{\mathrm{f}} x_{\mathrm{f}}(t) + B_{\mathrm{f}} s(t) + F_{\mathrm{f}} w_{\mathrm{g}}(t) \tag{6.3}$$

where $s = (s_a, s_b, s_c) \in \{-1,0,1\}^3$ is the vector of discrete-valued converter switch signals and the vector $w_{\mathrm{g}abc} = (w_{\mathrm{g}a}, w_{\mathrm{g}b}, w_{\mathrm{g}c}) \in \mathbb{R}^3$ represents the grid voltage. The system matrices are obtained by applying Kirchhoff's circuit laws and have the form

$$A_{\mathrm{f}} = \begin{bmatrix} -\frac{R_{\mathrm{fi}}}{L_{\mathrm{fi}}}\mathrm{I}_{3\times3} & 0_{3\times3} & -\frac{1}{L_{\mathrm{fi}}}\mathrm{I}_{3\times3} \\ 0_{3\times3} & -\frac{R_{\mathrm{fg}}}{L_{\mathrm{fg}}}\mathrm{I}_{3\times3} & \frac{1}{L_{\mathrm{fg}}}\mathrm{I}_{3\times3} \\ \frac{1}{C_{\mathrm{f}}}\mathrm{I}_{3\times3} & -\frac{1}{C_{\mathrm{f}}}\mathrm{I}_{3\times3} & 0_{3\times3} \end{bmatrix}, \quad B_{\mathrm{f}} = \frac{1}{L_{\mathrm{fi}}}\frac{1}{3}\begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \\ 0_{6\times1} & 0_{6\times1} & 0_{6\times1} \end{bmatrix}\frac{v_{\mathrm{dc}}}{2},$$

$$F_{\mathrm{f}} = \begin{bmatrix} 0_{3\times3} \\ -\frac{1}{L_{\mathrm{fg}}}\mathrm{I}_{3\times3} \\ 0_{3\times3} \end{bmatrix},$$

where the parameters correspond to the elements as illustrated in Fig. 6.1.

The three-phase grid voltage $w_{\mathrm{g}abc} = (w_{\mathrm{g}a}, w_{\mathrm{g}b}, w_{\mathrm{g}c})$ is assumed (I) sinusoidal and (II) three-phase symmetric:

$$\text{(I)} \quad w_{\mathrm{g}a}(t) = V_{\mathrm{g}} \sin(2\pi f_{\mathrm{g}} t + \varphi_{\mathrm{g}}), \tag{6.4}$$

$$\text{(II)} \quad w_{\mathrm{g}a}(t) = w_{\mathrm{g}b}(t - 1/(3f_{\mathrm{g}})) = w_{\mathrm{g}c}(t + 1/(3f_{\mathrm{g}})), \tag{6.5}$$

where V_g is the voltage amplitude, f_g is the grid frequency and φ_g is the initial phase of the grid. The sinusoidal three-phase symmetric voltage can be modeled in the $\alpha\beta$ coordiante system $w_{\mathrm{g}\alpha\beta}(t) = (w_{\mathrm{g}\alpha}(t), w_{\mathrm{g}\beta}(t))$ with

$$\dot{w}_{\mathrm{g}\alpha\beta}(t) = R_{2\times2}\, w_{\mathrm{g}\alpha\beta}(t), \qquad R_{2\times2} := \begin{bmatrix} 0 & -\omega_g \\ \omega_g & 0 \end{bmatrix}, \tag{6.6}$$

where $\omega_\mathrm{g} = 2\pi f_\mathrm{g}$ is the grid angular frequency. The three-phase voltage $w_{\mathrm{g}abc}(t)$ can then be obtained from $w_{\mathrm{g}\alpha\beta}(t)$ by using the inverse Clarke transform:

$$w_{\mathrm{g}abc}(t) = P_{3\times2}^{-1} w_{\mathrm{g}\alpha\beta}(t). \tag{6.7}$$

The above description leads to a state-space model involving the *LCL* electric circuit (6.3) and the grid voltage model (6.6) which takes the form

$$\dot{x}_\mathrm{s}(t) = A_\mathrm{s} x_\mathrm{s}(t) + B_\mathrm{s} s(t) \tag{6.8}$$

where

$$x_\mathrm{s} = \begin{bmatrix} x_\mathrm{f} \\ w_{\mathrm{g}\alpha\beta} \end{bmatrix}, \; A_\mathrm{s} = \begin{bmatrix} A_\mathrm{f} & F_\mathrm{f} P_{3\times2}^{-1} \\ 0_{2\times9} & R_{2\times2} \end{bmatrix}, \; B_\mathrm{s} = \begin{bmatrix} B_\mathrm{f} \\ 0_{2\times3} \end{bmatrix}. \tag{6.9}$$

6.3.3 State-space model of sinusoidal steady-state reference trajectory

The steady-state reference for each state of the *LCL* circuit (6.2) will be approximated by a sinusoidal shape. Assuming a three-phase symmetric and sinusoidal steady-state, the reference vector $x_{\mathrm{r}abc} \in \mathbb{R}^9$ can be modeled in $\alpha\beta$ coordinate frame by

$$\dot{x}_{\mathrm{r}\alpha\beta}(t) = R_{6\times6}\, x_{\mathrm{r}\alpha\beta}(t), \quad R_{6\times6} := \mathrm{blkdiag}(R_{2\times2}, R_{2\times2}, R_{2\times2}), \tag{6.10}$$

where

$$x_{\mathrm{r}\alpha\beta} = (i_{\mathrm{r,fi}\alpha}\,,\, i_{\mathrm{r,fi}\beta}\,,\, i_{\mathrm{r,fg}\alpha}\,,\, i_{\mathrm{r,fg}\beta}\,,\, v_{\mathrm{r,f}\alpha}\,,\, v_{\mathrm{r,f}\beta}) \tag{6.11}$$

consists of *LCL* state references in the $\alpha\beta$ frame, and $R_{2\times2}$ is the matrix defined in (6.6). To convert the references from the $\alpha\beta$ to the abc coordinate frame, one employs the inverse Clarke transform:

$$x_{\mathrm{r}abc}(t) = P_{9\times6}^{-1} x_{\mathrm{r}\alpha\beta}(t), \quad P_{9\times6}^{-1} := \mathrm{blkdiag}(P_{3\times2}^{-1}, P_{3\times2}^{-1}, P_{3\times2}^{-1}). \tag{6.12}$$

Obtaining the value of the state vector x_s in (6.8) at some t_0 can be done by using measured/estimated values of the components of x_s at time t_0. On the other hand, in order to obtain the reference trajectory value $x_{\mathrm{r}\alpha\beta}$ in (6.10) at some t_0, one needs to consider the desired active

and reactive power reference (P_g, Q_g) and the value of the grid voltage $w_{g\alpha\beta}(t_0)$. The mapping $x_{r\alpha\beta}(t_0) = f_r(P_g, Q_g, w_{g\alpha\beta}(t_0))$ can be derived by using Kirchhoff's laws and phasor notation, as described in what follows. Given the grid voltage phasor $\underline{\mathrm{V}}_\mathrm{g} = w_{g\alpha}(t_0) + j\, w_{g\beta}(t_0) = V_g e^{j\theta \mathrm{g}}$ and the desired active and reactive power (P_g, Q_g), the desired grid current has the value

$$\underline{\mathrm{I}}_\mathrm{fg} = I_\mathrm{fg} e^{j\left\{\arg\left(\underline{\mathrm{V}}_\mathrm{g}\right) - \phi_\mathrm{vi}\right\}}, \qquad I_\mathrm{fg} = \frac{1}{3V_\mathrm{g}}\sqrt{P_\mathrm{g}^2 + Q_\mathrm{g}^2}, \qquad \phi_\mathrm{vi} = \operatorname{atan}\left(\frac{Q_g}{P_g}\right), \tag{6.13}$$

with $\phi_\mathrm{vi} = \frac{\pi}{2}\operatorname{sign}(Q_g)$ in case of $P_g = 0$. The desired capacitor voltage and inverter current are

$$\underline{\mathrm{V}}_\mathrm{f} = \underline{\mathrm{V}}_\mathrm{g} + j\omega_\mathrm{g} L_\mathrm{fg} \underline{\mathrm{I}}_\mathrm{fg}, \qquad \underline{\mathrm{I}}_\mathrm{fi} = j\omega_\mathrm{g} C_\mathrm{f} \underline{\mathrm{V}}_\mathrm{f} + \underline{\mathrm{I}}_\mathrm{fg}. \tag{6.14}$$

The α and β components of the subvectors in $x_{r\alpha\beta}$ are then obtained by using the real and imaginary parts of the computed $\underline{\mathrm{I}}_\mathrm{fi}$, $\underline{\mathrm{I}}_\mathrm{fg}$ and $\underline{\mathrm{V}}_\mathrm{f}$:

$$i_{\mathrm{r,fi}\alpha}(t_0) = \mathrm{Re}\left\{\underline{\mathrm{I}}_\mathrm{fi}\right\}, \qquad i_{\mathrm{r,fi}\beta}(t_0) = \mathrm{Im}\left\{\underline{\mathrm{I}}_\mathrm{fi}\right\}, \tag{6.15a}$$

$$i_{\mathrm{r,fg}\alpha}(t_0) = \mathrm{Re}\left\{\underline{\mathrm{I}}_\mathrm{fg}\right\}, \qquad i_{\mathrm{r,fg}\beta}(t_0) = \mathrm{Im}\left\{\underline{\mathrm{I}}_\mathrm{fg}\right\}, \tag{6.15b}$$

$$v_{\mathrm{r,f}\alpha}(t_0) = \mathrm{Re}\left\{\underline{\mathrm{V}}_\mathrm{f}\right\}, \qquad v_{\mathrm{r,f}\beta}(t_0) = \mathrm{Im}\left\{\underline{\mathrm{V}}_\mathrm{f}\right\}. \tag{6.15c}$$

Note that the computed values also allow to obtain the desired steady-state sinusoidal voltage to be supplied by the inverter (the inverter's fundamental harmonic), which is given by the phasor

$$\underline{\mathrm{V}}_\mathrm{i} = \underline{\mathrm{V}}_\mathrm{f} + j\omega_\mathrm{g} L_\mathrm{fi} \underline{\mathrm{I}}_\mathrm{fi}. \tag{6.16}$$

6.3.4 Complete State-Space Model

The state-space model encompassing the system elements (NPC inverter, *LC* filter, power grid), as well as also sinusoidally approximated desired trajectories for the systems states, is obtained by combining the models of Sections 6.3.2 and 6.3.3 and has the form

$$\dot{x}(t) = Ax(t) + Bs(t), \tag{6.17a}$$

$$y(t) = Ex(t), \tag{6.17b}$$

where the matrices A, B, and E are given by

$$A = \begin{bmatrix} A_\mathrm{s} & 0_{11\times 6} \\ 0_{6\times 11} & R_{6\times 6} \end{bmatrix}, \quad B = \begin{bmatrix} B_\mathrm{s} \\ 0_{6\times 3} \end{bmatrix}, \quad E = \begin{bmatrix} I_{9\times 9} & 0_{9\times 2} & -P_{9\times 6}^{-1} \end{bmatrix}, \tag{6.18}$$

and $s(t)$, $y(t)$ and $x(t)$ are respectivelly the system input, output and state, with components as summarized below.

The system input $s \in \mathbb{R}^3$ with $s = (s_a, s_b, s_c)$ contains discrete-valued phase switching signals.

For the three-level NPC topology, the allowed values for each phase switching signal s_p, $p \in \{a,b,c\}$ are $s_p \in \{-1,0,1\}$.

The state vector $x \in \mathbb{R}^{17}$ consists of two subvectors, $x = (x_\mathrm{s}, x_{\mathrm{r}\alpha\beta})$ where $x_\mathrm{s} \in \mathbb{R}^{11}$ and $x_{\mathrm{r}\alpha\beta} \in \mathbb{R}^6$. The x_s contains as its subcomponents $x_\mathrm{s} = (i_{\mathrm{fi},abc}, i_{\mathrm{fg},abc}, v_{\mathrm{f},abc}, w_{\mathrm{g}\alpha\beta})$ where $i_{\mathrm{fi},abc} \in \mathbb{R}^3$ is the inverter-side current, $i_{\mathrm{fg},abc} \in \mathbb{R}^3$ is the grid-side current, $v_{\mathrm{f},abc} \in \mathbb{R}^3$ is the voltage of the *LC* filter capacitor, and $w_{\mathrm{g}\alpha\beta} \in \mathbb{R}^2$ is the grid voltage. On the other hand, the $x_{\mathrm{r}\alpha\beta}$ contains subcomponents $x_{\mathrm{r}\alpha\beta} = (i_{\mathrm{r,fi}\alpha\beta}, i_{\mathrm{r,fg}\alpha\beta}, v_{\mathrm{r,f}\alpha\beta})$ where $i_{\mathrm{r,fi}\alpha\beta} \in \mathbb{R}^2$, $i_{\mathrm{r,fg}\alpha\beta} \in \mathbb{R}^2$ and $v_{\mathrm{r,f}\alpha\beta} \in \mathbb{R}^2$ are sinusoidally approximated desired trajectories for the inverter-side current, grid-side current and capacitor voltage, respectively.

The output vector $y \in \mathbb{R}^9$ is a tracking-error vector, which as such consists of the difference between the *LC* circuit states $(i_{\mathrm{fi},abc}, i_{\mathrm{fg},abc}, v_{\mathrm{f},abc})$ and their sinusoidally approximated desired trajectories determined from the $x_{\mathrm{r}\alpha\beta}$.

For a given initial condition $x(t_0) = x_{t_0}$ and for some fixed value of the switching signal $s(t) = \ell$, $\forall\, t \geq t_0$, the solution of the state-space model (6.17) for time $t \geq t_0$ is

$$x(t) = e^{A(t-t_0)} x_{t_0} + \int_{t_0}^{t} e^{A(t-\tau)} B\ell \,\mathrm{d}\tau \tag{6.19}$$

which can compactly be written in the form

$$x(t) = C e^{\bar{A}_\ell (t-t_0)} \bar{x}_{t_0} \tag{6.20}$$

by introducing the definitions

$$C = \begin{bmatrix} I_{n\times n} & 0_{n\times 1} \end{bmatrix}, \quad \bar{A}_\ell = \begin{bmatrix} A & B\ell \\ 0_{1\times n} & 0_{1\times 1} \end{bmatrix}, \quad \bar{x}_{t_0} = \begin{bmatrix} x_{t_0} \\ 1 \end{bmatrix}.$$

It should be noted that for obtaining the state vector value $x = (x_\mathrm{s}, x_{\mathrm{r}\alpha\beta})$ at some time t_0, the subvector x_s can be formed by using the measured/estimated vector component values at t_0, and the subvector $x_{\mathrm{r}\alpha\beta}$ can be computed based on the desired active and reactive power reference (P_g, Q_g) and the value of the grid voltage $w_{\mathrm{g}\alpha\beta}(t_0)$, as described in Section 6.3.3.

6.4 Pulse Patterns Optimal Control Problem

The performance of power converter controllers is generally assessed based on the controller's ability to quickly reach a specified power reference (controller's transient performance) and based on the amount of higher harmonics of the current injected into the grid (controller's steady-state performance). In the case of medium-voltage power electronics applications, determining the best steady-state command input is a nonconvex optimization problem whose solving is too demanding to be performed online. Offline computed optimal steady-state command signals are referred to in the power electronics literature as Optimized Pulse

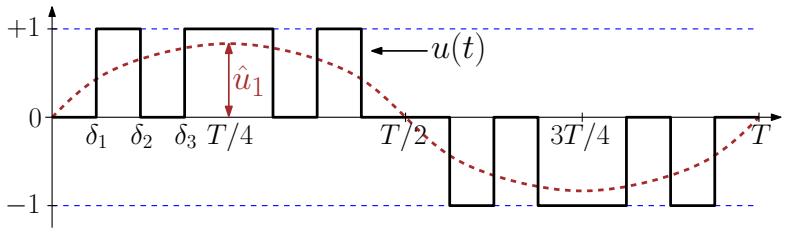


Figure 6.2 – An illustration of a three-level OPP signal $u(t)$ with $d = 3$ switching times over the first quarter of the period. The d switching times δ_1, δ_2 and δ_3 determine the shape of the OPP over the whole period T due to the quarter-wave symmetry imposed during OPP computation. The OPP is characterized by its fundamental harmonic amplitude $\hat{u}_1$.

Patterns (OPPs). Since OPPs will be used in the development of this chapter, an overview of their basic properties is provided in the next subsection, after which an optimal control problem (OCP) using OPPs is formulated.

6.4.1 Optimal Steady-State Operation: Optimized Pulse Patterns

In the power electronics literature, OPPs are generally considered as a mean whose utilization provides the best possible steady-state performance. An illustration of one period of an OPP signal is given in Fig. 6.2, where the OPP signal is denoted with $u(t)$ and its first (fundamental) harmonic amplitude with $\hat{u}_1$. A common assumption in the computation of OPPs is quarter-wave symmetry, whose consequence is that an OPP is completely determined by the switching times over the first quarter of its fundamental harmonic's period; see Fig. 6.2. The OPPs are computed for a fixed switching frequency, which is equivalent to fixing the number of available switching times within the first quarter of the period. This number is in the OPP literature called the pulse number, and will be denoted with d. The vector of d switching times over the first quarter of the period will be denoted as $\delta = (\delta_1, ..., \delta_d)$.

The steady-state performance metric is the total harmonic distortion of the current injected into the grid, which is a quantity proportional to a weighted sum of the squared OPP's voltage harmonics:

$$J_{opp}(\delta) = \sum_{i \in \Theta_d} w_i \, \hat{u}_i^2(\delta), \tag{6.21}$$

where w_i are the weighting coefficients, $\hat{u}_i$ are the magnitudes of the OPP's harmonics at frequencies $f_i = i/T$, and Θ_d denotes the set of indices of differential-mode harmonics, that is, of non-triple odd harmonic numbers excluding number one.

For a specified desired magnitude of the fundamental harmonic m, which is in the OPP literature also known as a modulation index, the OPP nonlinear optimization problem takes

the form:

$$\begin{aligned} \underset{\delta}{\text{minimize}} \quad & J_{opp}(\delta) \\ \text{subject to} \quad & \hat{u}_1(\delta) = m, \\ & \hat{u}_k(\delta) = 0, \ \ \forall k \in \Theta_e, \\ & 0 \leq \delta_1 \leq \ldots \leq \delta_d \leq T/4, \end{aligned} \tag{6.22}$$

where Θ_e is the set of harmonics to be eliminated, and $T/4$ is a quarter of the fundamental period. For a derivation of the harmonic magnitudes expressions $\hat{u}_k(\delta)$ and a detailed description of OPP computation, the reader is referred to the Section 3.4 of [34] and the references therein.

To allow an operation with various values of the fundamental harmonics, a common practice is that the modulation indiex values m are gridded with a fine stepsize over a certain range of interest $[m_{min}, m_{max}]$, the optimization problem (6.22) is solved for each of the gridding values of m, and the obtained optimal switching times $\delta^*(m)$, which are now a function of the modulation index, are stored in memory of the controller for online use. While the description given in this section corresponds to the three-level NPC topology of the present case study, the computation can also be generalized to topologies with higher numbers of levels. Furthermore, in the case of a grid-tied converter with *LC* filter, OPP computation also allows a shaping of the spectral content so that the size and weight of the *LC* filter are minimized. For these, and many other OPP related discussions, the reader is referred to [34].

6.4.2 Power Converter OCP based on Optimized Pulse Patterns

The control algorithm is executed with a control period T_s. It is assumed that at the beginning of each control period, perfect information of the system states is available and the control input is computed and applied immediately (i.e., computation delay is neglected). In practice, the computation delay of one control period can be approximately compensated by rotating the measured current and voltage vectors in the $\alpha\beta$ frame by angle $\omega_g T_s$ in the mathematically positive direction, as is done in step one of the control method from [36].

Given an active and reactive power reference pair (P_g, Q_g) and the grid voltage value $w_{g\alpha\beta}$ at the current time instant, the fundamental voltage harmonic desired to be applied by the inverter in steady-state is determined by the expression (6.16). This desired fundamental harmonic determines the modulation index of the OPP which should be applied in eventual steady-state operation; see Fig. 6.3 for an illustration.

The steady-state OPP will be used as a baseline input signal for transient optimization by taking a certain number of OPP switching times to form the controller's prediction horizon and adjusting them to reach the desired steady-state operation with specified (P_g, Q_g), as described in what follows. For a specified number of switching times N_{sw} to be considered within the controller's prediction horizon as decision variables, each of the three phases will

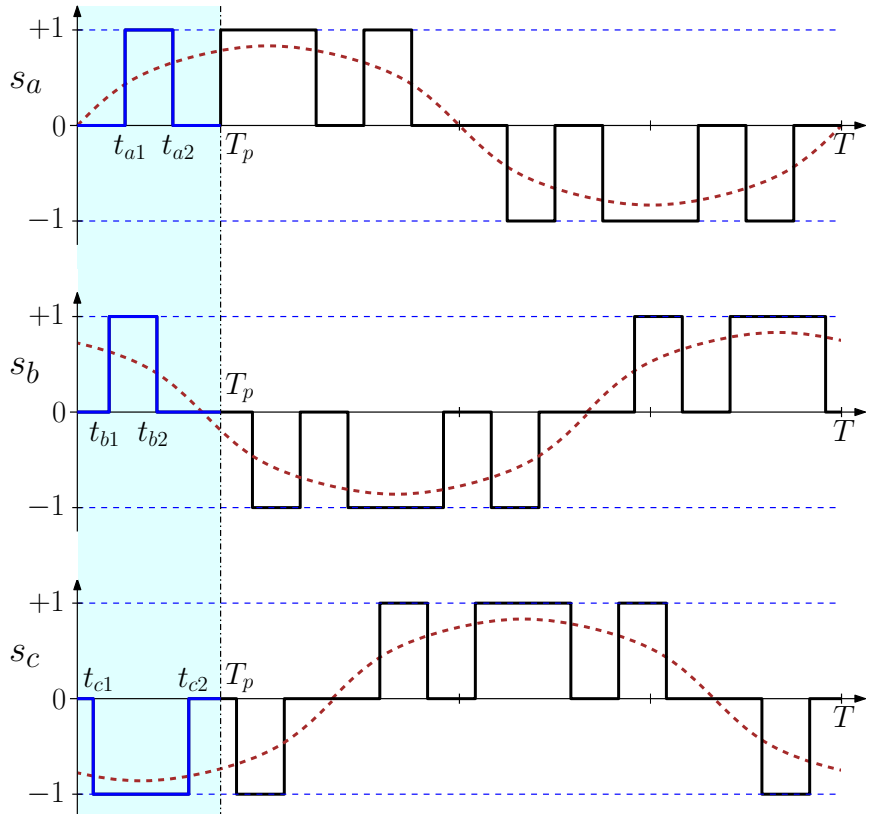


Figure 6.3 – Based on the desired active-reactive power reference (P_g, Q_g) and grid voltage value, the fundamental inverter harmonic to be generated in steady-state is determined, and is illustrated in the figure by dashed sinusoids in three phases. These fundamental harmonics determine the modulation index of the OPP in the figure. An example of a prediction horizon involving $N_{sw} = 6$ switching times is designated by the shaded region over the time interval $[0, T_p]$, with the N_{sw} switching times being the t_{a1}, t_{a2}, t_{b1}, t_{b2}, t_{c1} and t_{c2}. The value of T_p is selected to be the seventh OPP switching time from the beginning of the prediction horizon.

have some number of switching times taken from the OPP within the prediction horizon, as illustrated in Fig. 6.3. An illustration of the notation over the prediction horizon which will be introduced below is given in Fig. 6.4. Since the plant model is time-invariant, the time at the beginning of the controller's prediction horizon will be set to zero. The number of obtained switching times within phases a, b and c will be respectively denoted as n_a, n_b and n_c, where $N_{sw} = n_a + n_b + n_c$ holds. For each phase $i \in \{a, b, c\}$, the switching times of phase i are collected in a vector

$$t_i = (t_{i1}, \ldots, t_{in_i}), \quad i \in \{a, b, c\}, \tag{6.23}$$

with $t_i \in \mathbb{R}^{n_i}$. The vector of all switching times within the prediction horizon $\bar{t} \in \mathbb{R}^{N_{sw}}$ is thus

$$\bar{t} = (t_a, t_b, t_c). \tag{6.24}$$

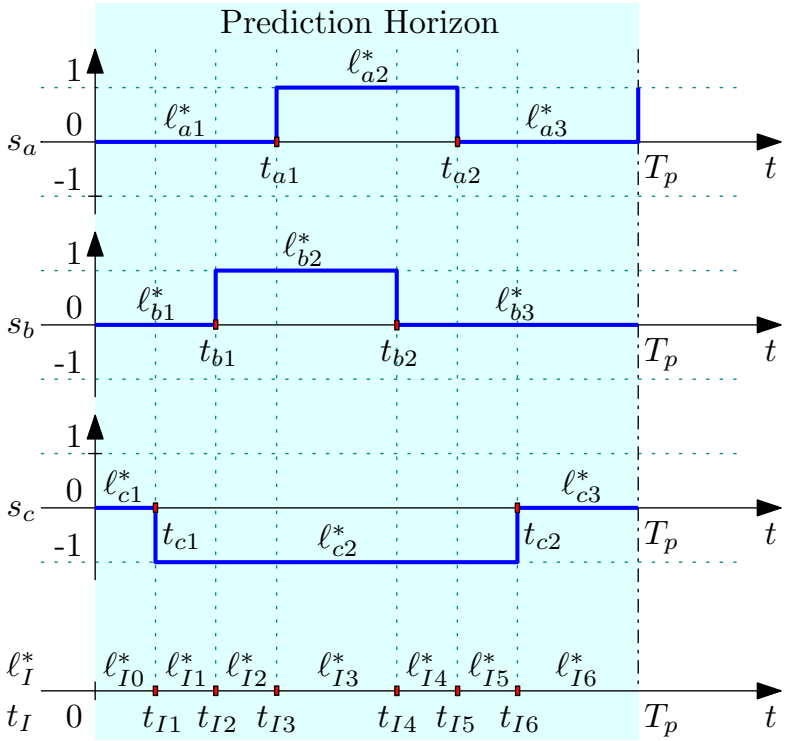


Figure 6.4 – An illustration of the switching signal $s(t)$ over a prediction horizon involving $N_{\text{sw}} = 6$. In the illustration, the switching times of each phase are $t_a = (t_{a1}, t_{a2})$, $t_b = (t_{b1}, t_{b2})$ and $t_c = (t_{c1}, t_{c2})$, resulting in a decision vector $\bar{t} = (t_a, t_b, t_c) = (t_{a1}, t_{a2}, t_{b1}, t_{b2}, t_{c1}, t_{c2})$. The switching values in each phase are $\ell_a^* = (0,1,0)$, $\ell_b^* = (0,1,0)$ and $\ell_c^* = (0,-1,0)$. The vector of three-phase switching times $\bar{t}$ is sorted in ascending order in vector $t_I = (0, \text{sort}(\bar{t}), T_p) = (0, t_{I1}, t_{I2}, t_{I3}, t_{I4}, t_{I5}, t_{I6}, T_p)$ where $t_{I1} = t_{c1}$, $t_{I2} = t_{b1}$, $t_{I3} = t_{a1}$, $t_{I4} = t_{b2}$, $t_{I5} = t_{a2}$, $t_{I6} = t_{c2}$. The three-phase switching values between pairs of subsequent times in t_I are $\ell_{I0}^* = (0,0,0)$, $\ell_{I1}^* = (0,0,-1)$, $\ell_{I2}^* = (0,1,-1)$, $\ell_{I3}^* = (1,1,-1)$, $\ell_{I4}^* = (1,0,-1)$, $\ell_{I5}^* = (0,0,-1)$, $\ell_{I6}^* = (0,0,0)$, which can be collected into a matrix $\ell_I^* = [\ell_{I0}^*, \ell_{I1}^*, \ell_{I2}^*, \ell_{I3}^*, \ell_{I4}^*, \ell_{I5}^*, \ell_{I6}^*]$.

While $\bar{t}$ will be a decision vector for the OCP, the star in the superscript $\bar{t}^*$ will denote the value of $\bar{t}$ corresponding to the OPP switching times within the prediction horizon, which will in what follows also be referred to as nominal (OPP) switching times. Furthermore, the values of the switching signals for each phase $i \in \{a, b, c\}$ which are obtained from the OPP are denoted with

$$\ell_i^* = (\ell_{i1}^*, \ldots, \ell_{i(n_i+1)}^*), \quad i \in \{a, b, c\}, \tag{6.25}$$

with $\ell_i^* \in \mathbb{R}^{n_i+1}$, where the indexing is such that ℓ_{ij}^* for all $j \in \{1, \ldots, n_i\}$ denotes the switching value from OPP existing before the time t_{ij}^*, and the last $\ell_{i(n_i+1)}^*$ is the switching value after the last time $t_{in_i}^*$ within the prediction horizon. For shorter notation, $\bar{\ell}^* = (\ell_a^*, \ell_b^*, \ell_c^*)$ is defined. The above definitions give rise to the input parametrization of $s(t)$ over the prediction horizion

$[0, T_p]$ which will be denoted with

$$\tilde{s}(t; \bar{t}, \bar{\ell}^*), \quad t \in [0, T_p], \tag{6.26}$$

and in case of $\tilde{s}(t; \bar{t}^*, \bar{\ell}^*)$ it represents the unmodified nominal OPP over the prediction horizon.

To set up the OCP for a certain power reference (P_g, Q_g) at some time instant t_0, the OCP's initial state vector x_{t_0} is formed from the measured/estimated states and the computed initialization of the reference trajectory as described in Section 6.3.3. To obtain the OPP nominal times $\bar{t}^*$ and switching values $\bar{\ell}^*$ within the prediction horizon, the desired value of the inverter's steady-state fundamental harmonic amplitude and angle is used (as computed in Section 6.3.3), yielding also the desired OPP's modulation index given by the expression

$$m = \frac{2}{v_{\mathrm{dc}}} V_i \tag{6.27}$$

where V_i is the amplitude of the desired inverter's steady-state fundamental harmonic. As the OPP is computed by doing a fine gridding over the range of interest for the modulation index values, the computed value m is rounded to the first available one without a significant impact. The obtained $\bar{t}^*$ and $\bar{\ell}^*$ are fully determining the shape of the input parametrization over the prediction horizon (6.26), and the first switching time in the OPP after the last one within $\bar{t}^*$ is taken to be the prediction horizon length T_p, which is thus a horizon length that takes a different value at each new control period. By defining a cost term representing the integral of the squared tracking error:

$$J_{te}(\bar{t}) = \int_0^{T_p} y(t)^T Q y(t) \mathrm{d}t, \tag{6.28}$$

the penalization of the devitaion of the switching times $\bar{t}$ from the nominal $\bar{t}^*$ switching times:

$$J_{ds}(\bar{t}; \bar{t}^*) = \| \bar{t} - \bar{t}^* \|_2^2, \tag{6.29}$$

and a polytopic constraint set preserving the sequence of the switching times t_i, $i \in \{a, b, c\}$ within each phase:

$$\Xi_{ps} = \left\{ \bar{t} \mid \forall i \in \{a, b, c\} \text{ and } \forall j \in \{2, ..., n_i\},\ 0 \leq t_{i1},\ t_{i(j-1)} \leq t_{ij},\ t_{in_i} \leq T_p \right\}, \tag{6.30}$$

the OCP takes the form:

$$\begin{aligned}
\underset{\bar{t}, x(\cdot), y(\cdot)}{\text{minimize}} \quad & J_{te}(\bar{t}) + J_{ds}(\bar{t}; \bar{t}^*) \\
\text{subject to} \quad & x(0) = x_{t_0}, \\
& \dot{x}(t) = Ax(t) + B\tilde{s}(t; \bar{t}, \bar{\ell}^*), \quad t \in [0, T_p], \\
& y(t) = Ex(t), \qquad\qquad\quad t \in [0, T_p], \\
& \bar{t} \in \Xi_{ps}.
\end{aligned} \tag{6.31}$$

The optimal solution of the OCP, which is the input waveform $\tilde{s}(t;\bar{t},\bar{\ell}^*)$ involving the computed optimal $\bar{t}$, is applied by the converter over the control period $[0, T_s]$ (the control period T_s is much shorter than the prediction horizon T_p over which the input $\tilde{s}$ was optimized), and the process is repeated at the beginning of the next control period.

Besides the finiteness of the horizon length T_p in (6.31), which is an obvious source of suboptimality in the obtained control law, the stated OCP also does not involve the sequence of switching levels over the prediction horizion (the $\bar{\ell}^*$) among its decision variables. The fixing of the sequence of switching levels to the sequence existing in OPP removes the combinatorial nature from the optimization problem (6.31), making it considerably easier to solve. Furthermore, there are also possibilities to reduce this source of suboptimality by employing the pulse insertion techniques that already exist for other familiar methods; for description, the reader is referred to Section 12.6 of [34].

6.4.3 Sequential Approach: Elimination of System Dynamics

One way to solve the OCP (6.31) is by substituting the system dynamics into the cost function in a manner of the sequential approach to optimal control, resulting in an optimization problem with a finite number of decision variables. In particular, given a vector $\bar{t}$, by sorting its elements in an ascending order (with the operator sort used below), a vector-valued mapping t_I is introduced:

$$t_I(\bar{t}) = (0, \mathrm{sort}(\bar{t}), T_p) = (0, t_{I1}, t_{I2}, \ldots, t_{IN_{sw}}, T_p), \tag{6.32}$$

where $t_I : \mathbb{R}^{N_{sw}} \to \mathbb{R}^{N_{sw}+2}$ and in which the first and the last element of t_I are constants appended for later notational convenience (they will also be referred to as t_{I0} and $t_{I(N_{sw}+1)}$). The prediction horizon can then be divided into subintervals:

$$[0, T_p] = [0, t_{I1}] \cup [t_{I1}, t_{I2}] \cup \cdots \cup [t_{IN_{\mathrm{sw}}}, T_p], \tag{6.33}$$

where over each subinterval $[t_{Ik}, t_{I(k+1)}]$ with $k \in \{0, \ldots, N_{sw}\}$ there is a three-phase switching value $\ell^*_{Ik} \in \mathbb{R}^3$ corresponding to the input parametrization (6.26); see Fig. 6.4 for an illustration. These switching values ℓ^*_{Ik} are uniquelly determined by $\tilde{s}(t;\bar{t},\bar{\ell}^*)$ and form a sequence of three-phase switching values over the prediction horizon, which is a matrix-valued mapping ℓ^*_I defined as:

$$\ell^*_I(\bar{t}) = [\ell^*_{I0}, \ell^*_{I1}, \ldots, \ell^*_{IN_{sw}}] \tag{6.34}$$

where $\ell^*_I : \mathbb{R}^{N_{sw}} \to \mathbb{R}^{3\times(N_{sw}+1)}$. Note that, based on the input parametrization (6.26), the ℓ^*_I depends on the switching times $\bar{t}$ since ℓ^*_I changes whenever any two switching times of two different phases change their order with respect to each other. By using the above notation, for any given $\bar{t}$ one can compute the cost term $J_{te}(\bar{t})$ by introducing into it the system

dynamics (6.17) as follows:

$$\begin{aligned}
J_{te}(\bar{t}) &= \int_0^{T_p} y(t)^T Q y(t)\mathrm{d}t = \sum_{k=0}^{N_{sw}} \int_{t_{I_k}}^{t_{I_{k+1}}} y(t)^T Q y(t)\mathrm{d}t \\
&= \sum_{k=0}^{N_{sw}} \int_{t_{I_k}}^{t_{I_{k+1}}} \big(Ex(t)\big)^T Q\big(Ex(t)\big)\mathrm{d}t \\
&= \sum_{k=0}^{N_{sw}} \int_{t_{I_k}}^{t_{I_{k+1}}} \big(EC e^{\bar{A}_{\ell^*_{Ik}}(t-t_{I_k})}\bar{x}_k\big)^T Q\big(EC e^{\bar{A}_{\ell^*_{Ik}}(t-t_{I_k})}\bar{x}_k\big)\mathrm{d}t \\
&= \sum_{k=0}^{N_{sw}} \bar{x}_k^T \int_{t_{I_k}}^{t_{I_{k+1}}} e^{\bar{A}^T_{\ell^*_{Ik}}(t-t_{I_k})}\big(C^T E^T QCE\big)e^{\bar{A}_{\ell^*_{Ik}}(t-t_{I_k})}\mathrm{d}t\,\bar{x}_k \\
&= \sum_{k=0}^{N_{sw}} \bar{x}_k^T\, N_k\, \bar{x}_k,
\end{aligned} \tag{6.35}$$

where the equality at line two uses (6.17b), the equality at line three uses the state-space solution (6.20) with $\bar{x}_k = (x(t_{Ik}), 1)$ representing the values of state vectors (appended with one) at the edges of the subintervals (6.33), and line five introduces N_k to denote the integral from line four:

$$N_k = \int_{t_{I_k}}^{t_{I_{k+1}}} e^{\bar{A}^T_{\ell^*_{Ik}}(t-t_{I_k})}\big(C^T E^T QCE\big)e^{\bar{A}_{\ell^*_{Ik}}(t-t_{I_k})}\mathrm{d}t.$$

The value of the integral can be computed by applying the results of [47] yielding

$$N_k = F_{k3}^T G_{k2}, \quad k \in \{0, \ldots, N_{sw}\}, \tag{6.36}$$

where F_{k3} and G_{k2} are submatrices of the matrix exponential

$$e^{\hat{A}_{\ell^*_{Ik}}(t_{I_{k+1}}-t_{I_k})} = \begin{bmatrix} F_{k2} & G_{k2} \\ 0_{(n+1)\times(n+1)} & F_{k3} \end{bmatrix} \tag{6.37}$$

which is formed by using

$$\hat{A}_{\ell^*_{Ik}} = \begin{bmatrix} -\bar{A}^T_{\ell^*_{Ik}} & C^T E^T QEC \\ 0_{(n+1)\times(n+1)} & \bar{A}_{\ell^*_{Ik}} \end{bmatrix}. \tag{6.38}$$

For a given initial state value x_{t_0}, the optimization problem (6.31) thus takes the form involving only $\bar{t}$ as a decision vector:

$$\begin{aligned}
&\underset{\bar{t}}{\text{minimize}} \quad \sum_{k=0}^{N_{sw}} \bar{x}_k^T(\bar{t})\, N_k(\bar{t})\, \bar{x}_k(\bar{t}) + J_{ds}(\bar{t}; \bar{t}^*) \\
&\text{subject to} \quad \bar{t} \in \Xi_{ps}.
\end{aligned} \tag{6.39}$$

where the matrix-valued mappings $N_k(\bar{t})$, $k \in \{0, 1, ..., N_{sw}\}$ are defined by (6.36), (6.37) and

(6.38) in which the $t_I(\bar{t})$ and $\ell_I^*(\bar{t})$ that correspond to a selected $\bar{t}$ are used, and the $\bar{x}_k(\bar{t})$ can be evaluated by using $\bar{x}_0 = (x_{t_0}, 1)$ and recursively applying the equation (6.20) as

$$\bar{x}_k(\bar{t}) = e^{\bar{A}_{\ell^*_{I(k-1)}}(t_{I(k)} - t_{I(k-1)})} \bar{x}_{k-1}(\bar{t}) \tag{6.40}$$

with $t_I(\bar{t})$ and $\ell_I^*(\bar{t})$ again determined based on $\bar{t}$. The obtained optimization problem can be tackled for example by applying a gradient projection method, as described in the following subsection.

6.4.4 Solution Approach based on Gradient Projection

By denoting the cost function in (6.39) by

$$J_{tot}(\bar{t}) = \sum_{k=0}^{N_{sw}} \bar{x}_k^T(\bar{t})\ N_k(\bar{t})\ \bar{x}_k(\bar{t}) + J_{ds}(\bar{t}; \bar{t}^*) = J_{te}(\bar{t}) + J_{ds}(\bar{t}; \bar{t}^*), \tag{6.41}$$

given an initial iterate $\bar{t}^0 \in \mathbb{R}^{N_{sw}}$, the gradient projection method [11] for (6.39) takes the form:

$$\bar{t}^{q+1} = P_{\Xi_{ps}}\left(\bar{t}^q - s^q_{\text{gp}} \nabla J_{tot}(\bar{t}^q)\right), \tag{6.42}$$

where $\bar{t}^q \in \mathbb{R}^{N_{sw}}$ is the vector of switching times at the q-th iteration of the gradient projection algorithm, $P_{\Xi_{ps}} : \mathbb{R}^{N_{sw}} \to \mathbb{R}^{N_{sw}}$ denotes the projection on the set Ξ_{ps} as defined in (6.30), and $s^q_{\text{gp}} > 0$ is the stepsize at the iteration q chosen so that the following condition holds:

$$J(\bar{t}^q) - J(\bar{t}^{q+1}) \geq \sigma \nabla J(\bar{t}^q)^T \left(\bar{t}^q - \bar{t}^{q+1}\right), \tag{6.43}$$

with $\sigma \in (0,1)$. The stepsize s^q_{gp} satisfying (6.43) at iteration q can be found by using backtracking, i.e., by examining for some fixed $\beta \in (0,1)$ and $s_{\text{init}} > 0$ the sequence of values $\{s_{\text{init}}, \beta s_{\text{init}}, \beta^2 s_{\text{init}}, \beta^3 s_{\text{init}}, ...\}$ and taking as s^q_{gp} the largest one for which (6.43) holds. An appropriate choice of the initial iterate $\bar{t}^0$ can be obtained by using the vector of nominal switching times $\bar{t}^*$.

The gradient projection iteration (6.42) requires the gradient $\nabla J_{tot}(\bar{t}^q)$. The partial derivatives of $J_{tot}(\bar{t})$ with respect to the components of $\bar{t}$ are more easily stated with respect to the corresponding components of the sorted vector t_I defined in (6.32). By using the chain rule, the partial derivative of the $J_{te}(\bar{t})$ part of $J_{tot}(\bar{t})$ with respect to t_{Ij}, $j \in \{1, \ldots, N_{sw}\}$ is given by

$$\frac{\partial J_{te}(\bar{t})}{\partial t_{Ij}} = \sum_{k=0}^{N_{sw}} \left(\frac{\partial \bar{x}_k^T(\bar{t})}{\partial t_{Ij}}\ N_k(\bar{t})\ \bar{x}_k(\bar{t}) + \bar{x}_k^T(\bar{t})\ \frac{\partial N_k(\bar{t})}{\partial t_{Ij}}\ \bar{x}_k(\bar{t}) + \bar{x}_k^T(\bar{t})\ N_k(\bar{t})\ \frac{\partial \bar{x}_k(\bar{t})}{\partial t_{Ij}} \right). \tag{6.44}$$

The gradient expression (6.44) involves the partial derivatives $\partial \bar{x}_k / \partial t_{Ij}$ and $\partial N_k / \partial t_{Ij}$ whose expressions will be derived now. The expressions for the partial derivatives $\partial \bar{x}_k / \partial t_{Ij}$ are obtained from (6.40) and have the following form:

- For $k > j+1$

$$\frac{\partial \bar{x}_k(\bar{t})}{\partial t_{Ij}} = \prod_{m=j+1}^{k-1} e^{\bar{A}_{\ell^*_{Im}}(t_{I(m+1)}-t_{Im})}(-\bar{A}_{\ell^*_{Ij}}) \prod_{m=0}^{j} e^{\bar{A}_{\ell^*_{Im}}(t_{I(m+1)}-t_{Im})}\bar{x}_0 + \prod_{m=j}^{k-1} e^{\bar{A}_{\ell^*_{Im}}(t_{I(m+1)}-t_{Im})}\bar{A}_{\ell^*_{I(j-1)}} \prod_{m=0}^{j-1} e^{\bar{A}_{\ell^*_{Im}}(t_{I(m+1)}-t_{Im})}\bar{x}_0. \tag{6.45}$$

- For $k = j+1$

$$\frac{\partial \bar{x}_k(\bar{t})}{\partial t_{Ij}} = (-\bar{A}_{\ell^*_{Ij}}) \prod_{m=0}^{j} e^{\bar{A}_{\ell^*_{Im}}(t_{I(m+1)}-t_{Im})}\bar{x}_0 + e^{\bar{A}_{\ell^*_{Ij}}(t_{I(j+1)}-t_{Ij})}\bar{A}_{\ell^*_{I(j-1)}} \prod_{m=0}^{j-1} e^{\bar{A}_{\ell^*_{Im}}(t_{I(m+1)}-t_{Im})}\bar{x}_0. \tag{6.46}$$

- For $k = j$

$$\frac{\partial \bar{x}_k(\bar{t})}{\partial t_{Ij}} = \bar{A}_{\ell^*_{I(k-1)}} \prod_{m=0}^{k-1} e^{\bar{A}_{\ell^*_{Im}}(t_{I(m+1)}-t_{Im})}\bar{x}_0. \tag{6.47}$$

- For $k < j$

$$\frac{\partial \bar{x}_k(\bar{t})}{\partial t_{Ij}} = 0. \tag{6.48}$$

The expressions for the partial derivatives $\partial N_k/\partial t_{Ij}$ are obtained from (6.36)-(6.37). By using (6.37), the N_k in (6.36) can be expressed as

$$N_k(\bar{t}) = \begin{bmatrix} 0 & I \end{bmatrix} e^{\hat{A}^T_{\ell^*_{Ik}}(t_{I(k+1)}-t_{Ik})} \begin{bmatrix} 0 & 0 \\ I & 0 \end{bmatrix} e^{\hat{A}_{\ell^*_{Ik}}(t_{I(k+1)}-t_{Ik})} \begin{bmatrix} 0 \\ I \end{bmatrix}, \tag{6.49}$$

where the I and 0 respectively denote $I_{(n+1)\times(n+1)}$ and $0_{(n+1)\times(n+1)}$ with $n = 17$ being the order of the state-space model (6.17). By introducing

$$T_1 := \begin{bmatrix} 0 & I \end{bmatrix}, \quad T_2 := \begin{bmatrix} 0 & 0 \\ I & 0 \end{bmatrix}, \quad T_3 := \begin{bmatrix} 0 \\ I \end{bmatrix}, \tag{6.50}$$

the partial derivatives of N_k matrices are:

- For $k = j$

$$\frac{\partial N_k(\bar{t})}{\partial t_{Ij}} = T_1 \cdot \left(-\hat{A}^T_{\ell^*_{Ik}}\right) \cdot e^{\hat{A}^T_{\ell^*_{Ik}}(t_{I(k+1)}-t_{Ik})} \cdot T_2 \cdot e^{\hat{A}_{\ell^*_{Ik}}(t_{I(k+1)}-t_{Ik})} \cdot T_3 + T_1 \cdot e^{\hat{A}^T_{\ell^*_{Ik}}(t_{I(k+1)}-t_{Ik})} \cdot T_2 \cdot \left(-\hat{A}_{\ell^*_{Ik}}\right) \cdot e^{\hat{A}_{\ell^*_{Ik}}(t_{I(k+1)}-t_{Ik})} \cdot T_3. \tag{6.51}$$

- For $k+1=j$

$$\frac{\partial N_k(\bar{t})}{\partial t_{Ij}} = T_1 \cdot \hat{A}^T_{\ell^*_{Ik}} \cdot e^{\hat{A}^T_{\ell^*_{Ik}}(t_{I(k+1)}-t_{Ik})} \cdot T_2 \cdot e^{\hat{A}_{\ell^*_{Ik}}(t_{I(k+1)}-t_{Ik})} \cdot T_3 + T_1 \cdot e^{\hat{A}^T_{\ell^*_{Ik}}(t_{I(k+1)}-t_{Ik})} \cdot T_2 \cdot \hat{A}_{\ell^*_{Ik}} \cdot e^{\hat{A}_{\ell^*_{Ik}}(t_{I(k+1)}-t_{Ik})} \cdot T_3. \tag{6.52}$$

- For $k < j-1$ or $k > j$

$$\frac{\partial N_k(\bar{t})}{\partial t_{Ij}} = 0_{(n+1)\times(n+1)}. \tag{6.53}$$

On the other hand, the partial derivatives of the term $J_{ds}(\bar{t};\bar{t}^*)$ in (6.41) with respect to t_{Ij}, $j \in \{1,\ldots,N_{sw}\}$ are given by

$$\frac{\partial J_{ds}(\bar{t};\bar{t}^*)}{\partial t_{Ij}} = 2\left(t_{Ij} - t^*_{Ij}\right). \tag{6.54}$$

where $J_{ds}(\bar{t};\bar{t}^*)$ is defined in (6.29).

Nevertheless, the amount of computation required for solving the problem (6.31) without additional approximations is not convenient for real-time execution on an embedded hardware. The following section describes approximations of the OCP (6.31) which will be introduced through the form (6.39) to obtain a low computational complexity approximation of the optimal control policy. The solutions of (6.31) obtained by applying the gradient projection algorithm of this section will be used for comparison to assess the influence which the introduced approximations have on the obtained performance.

6.5 Approximate Pulse Patterns Optimal Control

The starting point for obtaining a low computational complexity approximation of the OCP (6.31) is its QP approximation which will be described in this section. Consider the (nominal) OPP sequence of switching times $\bar{t}^*$ along the prediction horizon. It is uniquely determining its sorted version t^*_I (i.e., the $t_I(\bar{t}^*)$) and the corresponding three-phase switching values $\ell^*_I(\bar{t}^*)$. By using (6.20), one can perform a forward simulation of the plant dynamics for the OPP input (the $\tilde{s}(t;\bar{t}^*,\bar{\ell}^*)$) along the prediction horizon to obtain the values $\bar{x}_k$:

$$\bar{x}_k(\bar{t}^*) = e^{\bar{A}_{\ell^*_{I(k-1)}}(t^*_{I(k)}-t^*_{I(k-1)})}\bar{x}_{k-1}(\bar{t}^*), \tag{6.55}$$

where the index k in $\bar{x}_k$ denotes that it is the value at the k-th time instant (the instant t^*_{Ik}) with $k \in \{1,...,N_{sw}\}$ and the initial state is $\bar{x}_0 = (x_{t_0},1)$. For the values $\bar{x}_k$ obtained for $\bar{t}^*$, one can perform a linearization of $\bar{x}_k(\bar{t}^*)$ with respect to the time deviations Δt_I:

$$\tilde{x}_k(\Delta t_I;\bar{t}^*) = \bar{x}_k(\bar{t}^*) + \nabla\bar{x}_k(\bar{t}^*)^T\Delta t_I, \tag{6.56}$$

where the vector $\Delta t_I \in \mathbb{R}^{N_{sw}}$ is defined as

$$\Delta t_I = (t_{I1} - t^*_{I1}, \ldots, t_{IN_{sw}} - t^*_{IN_{sw}}), \tag{6.57}$$

the matrix $\nabla \bar{x}_k(\bar{t}^*)^T$ represents the Jacobian (i.e., transposed gradient) matrix:

$$\nabla \bar{x}_k(\bar{t}^*)^T = \left[\frac{\partial \bar{x}_k}{\partial t_{I1}}(\bar{t}^*), \ldots, \frac{\partial \bar{x}_k}{\partial t_{IN_{sw}}}(\bar{t}^*) \right]^T \tag{6.58}$$

where for each $i \in \{1, ..., N_{sw}\}$, the column-vector $\frac{\partial \bar{x}_k}{\partial t_{Ii}}(\bar{t}^*)$ contains partial derivatives of the components of $\bar{x}_k(\bar{t}^*)$ with respect to the i-th switching time t_{Ii}. The expressions for vectors $\frac{\partial \bar{x}_k}{\partial t_{Ii}}(\bar{t})$, $i \in \{1, ..., N_{sw}\}$ are given by the equations (6.45)-(6.48). Note that since the linearization (6.58) is performed at $\bar{t}^*$ which is characterized by its switching sequence $\ell^*_I(\bar{t}^*)$, the linearization is valid only as long as the switching times in different phases have not changed their order with respect to each other, or more formally, as long as there holds $\Delta t_I \in \Xi_s$ where

$$\begin{aligned} \Xi_s(\bar{t}^*) = \big\{ \Delta t_I \,|\, & 0 \le t^*_{I1} + \Delta t_{I1},\ t^*_{IN_{sw}} + \Delta t_{IN_{sw}} \le T_p, \\ & \forall i \in \{1, ..., N_{sw} - 1\},\ t^*_{Ii} + \Delta t_{Ii} \le t^*_{I(i+1)} + \Delta t_{I(i+1)} \big\}. \end{aligned} \tag{6.59}$$

It should also be noted that the vector Δt_I from (6.57) is of dimension N_{sw}, while t_I from (6.32) is of dimension $N_{sw} + 2$ due to the appended zero and T_p. For the sake of better readability, mathematical expressions of the form $t_I + \Delta t_I$ will be interpreted in what follows as a sum of the vector t_I and the vector Δt_I to which zeros are appended as the first and as the last component.

The QP approximation of (6.31) is obtained by substituting the linearized expressions $\tilde{x}_k(\Delta t_I; \bar{t}^*)$ from (6.58) in place of $\bar{x}_k$ in (6.39). The cost function then has the form

$$\sum_{k=0}^{N_{sw}} \tilde{x}_k(\Delta t_I; \bar{t}^*)^T N_k(\bar{t}^*)\, \tilde{x}_k(\Delta t_I; \bar{t}^*) + J_{\Delta ds}(\Delta t_I; \bar{t}^*),$$

where $J_{\Delta ds}(\Delta t_I; \bar{t}^*)$ is the cost term from (6.29) expressed as a function of Δt_I. By defining a (polytopic) box constraint set

$$\Xi_b(\delta_{max}) = \big\{ \Delta t_I \,|\, \forall i \in \{1, ..., N_{sw}\},\ |\Delta t_{Ii}| \le \delta_{max} \big\}$$

whose purpose is to keep the introduced linearizations $\tilde{x}_k(\Delta t_I; \bar{t}^*)$ valid by bounding the components of Δt_I, a QP of the following form is obtained:

$$\begin{aligned} \underset{\Delta t_I}{\text{minimize}}\ & \Delta t_I^T H(\bar{t}^*) \Delta t_I + h(\bar{t}^*) \Delta t_I + J_{\Delta ds}(\Delta t_I; \bar{t}^*) \\ \text{subject to}\ & \Delta t_I \in \Xi_b(\delta_{max}), \\ & \Delta t_I \in \Xi_s(t^*_I), \end{aligned} \tag{6.60}$$

where the Hessian $H(\bar{t})$, vector $h(\bar{t})$, and a constant scalar term $c(\bar{t})$ which is omitted from the

cost of (6.60) are having the forms

$$H(\bar{t}) = \sum_{k=0}^{N_{sw}} \nabla \bar{x}_k(\bar{t})^T N_k(\bar{t}) \nabla \bar{x}_k(\bar{t}),$$
$$h(\bar{t}) = \sum_{k=0}^{N_{sw}} \bar{x}_k(\bar{t})^T \left(N_k(\bar{t}) + N_k^T(\bar{t})\right) \bar{x}_k(\bar{t}),$$
$$c(\bar{t}) = \sum_{k=0}^{N_{sw}} \bar{x}_k(\bar{t})^T N_k(\bar{t}) \bar{x}_k(\bar{t}).$$

To make the approach based on the QP from (6.60) computationally convenient for potential real-time implementation, several additional aspects need to be considered. These aspects, which are related to simulation of the system dynamics and efficient obtaining of an approximate solution to the optimization problem, are addressed in the following subsections.

6.5.1 Dynamic Information: Precomputed Matrix Exponentials

Similarly to the idea of storing the offline computed OPPs in the controller's memory in order to obtain a high-performance steady-state operation without the online OPP computation, in order to perform prediction-based optimization of transients, the computational task of predicting the state values by the equation (6.55) can potentially be done by storing the matrix exponentials representing the system dynamics from expression (6.20) into the controller's memory. This is motivated by a finite number of voltage vectors that can be produced by the power converter, which are illustrated in Fig. 6.5 for the case of a three-level converter. By denoting with $\ell_{abc} = (\ell_a, \ell_b, \ell_c) \in \{-1, 0, 1\}^3$ the switching values (i.e., the state-space input signal values s) that are allowed by the converter, although the total number of different switching combinations is $3^3 = 27$, the actual total number of voltage vectors $v_{i\alpha\beta}$ that can be produced by the inverter:

$$v_{i\alpha\beta} = \frac{v_{\mathrm{dc}}}{2} P_{2\times 3}\, \ell_{abc} \tag{6.61}$$

is 19 (see Fig. 6.5). Denote with $\mathscr{L}_{abc}$ a (non-unique) set of 19 switching values ℓ_{abc} which cover all voltage vectors that can be produced. Any switching value ℓ_{abc} which is not in $\mathscr{L}_{abc}$ can always be mapped to at least one switching value in the set that produces the same voltage vector. For each switching value $\ell_{abc} \in \mathscr{L}_{abc}$, there is a corresponding matrix exponential $e^{\bar{A}_{\ell_{abc}}\tau}$ where the argument $\tau \geq 0$ can take only non-negative values. By considering a certain range of the argument values $[0, \tau_{max}]$, for each of the 19 switching values in $\mathscr{L}_{abc}$ (i.e., for each voltage vector that can be produced) one can perform a gridding of the range $[0, \tau_{max}]$ with n_g points $\mathscr{T}_g = \{\tau_{g0}, \tau_{g1}, ..., \tau_{gn_g}\}$ and store in the controller's memory the set of matrix exponentials:

$$\mathscr{S}_{\ell_{abc}}(\mathscr{T}_g) = \left\{ e^{\bar{A}_{\ell_{abc}}\tau} \mid \tau \in \mathscr{T}_g \right\}. \tag{6.62}$$

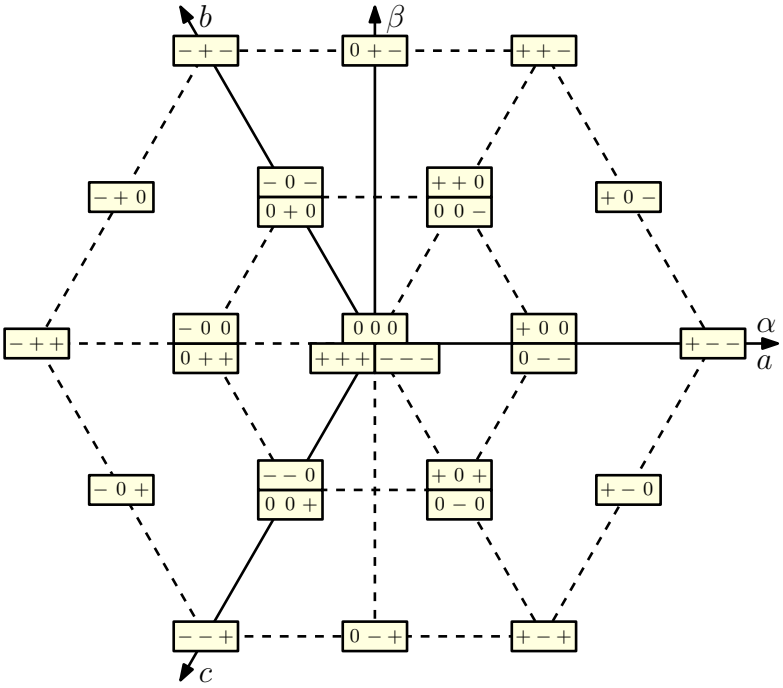


Figure 6.5 – Volage vectors of a three-phase three-level NPC inverter in $\alpha\beta$ frame. The switching value ℓ_{abc} of each vector is written in rectangle with plus and minus signs denoting the switching values one and negative one, respectively. Although the total number of rectangles (switching values) is 27, the total number of voltage vectors that inverter can produce is 19.

During online operation, for a given switching value ℓ_{abc} and a given argument value τ, it is possible to perform a convex combination of the stored matrix exponentials that correspond to the ℓ_{abc} by taking the gridding points adjacent to the argument τ. More formally, these are the points $\tau_{gi}, \tau_{g(i+1)} \in \mathcal{T}_g$ where i is such that $\tau_{gi} \leq \tau \leq \tau_{g(i+1)}$, and the evaluation of the matrix exponential $e^{\bar{A}_{\ell_{abc}}\tau}$ by forming a convex combination of the stored matrix exponentials takes the form:

$$e^{\bar{A}_{\ell_{abc}}\tau} \approx \frac{\tau_{g(i+1)}-\tau}{\tau_{g(i+1)}-\tau_{gi}} e^{\bar{A}_{\ell_{abc}}\tau_{gi}} + \frac{\tau-\tau_{gi}}{\tau_{g(i+1)}-\tau_{gi}} e^{\bar{A}_{\ell_{abc}}\tau_{g(i+1)}}.$$

Alternatively, one can simply round the value τ to the closest available one within the grid $\mathcal{T}_g$ and use its matrix exponential as an approximate evaluation. The memory-based evaluations of matrix exponentials $e^{\bar{A}_{\ell_{abc}}\tau}$ can be used both for the forward system simulation in (6.55) as well as for forming of the Jacobian in (6.56) which involves the sensitivities of the predicted states with respect to the switching times.

In addition to storing the matrix exponentials $\mathcal{S}_{\ell_{abc}}$ of the system dynamics, one can as well store the matrix exponentials $e^{\hat{A}_{\ell_{abc}}\tau}$ of (6.37) needed for calculating the N_k terms (6.36) in the cost function. Again, for selected gridding points $\mathcal{T}_g$ (which could possibly be different than

the ones used for $\mathscr{S}_{\ell_{abc}}$) and for the switching values $\ell_{abc} \in \mathscr{L}_{abc}$, one stores in memory

$$\mathscr{C}_{\ell_{abc}}(\mathscr{T}_g) = \left\{ e^{\hat{A}_{\ell_{abc}}\tau} \mid \tau \in \mathscr{T}_g \right\}, \tag{6.63}$$

which as well allows doing online convex combinations of the stored matrices or possibly rounding of τ to the closest gridding value in $\mathscr{T}_g$. In case of matrices of (6.37), one could alternatively also directly store the matrices N_k of (6.36) for the gridding points $\mathscr{T}_g$ and operate with them.

The memory requirements for storing of matrix exponentials can be reduced by exploiting their sparsity and special structure. For example, the matrix exponential of system dynamics (6.20) has a similar strucure for the switching values ℓ_{abc} that contain two zeros, such as $(1,0,0)$, $(0,1,0)$ or $(0,-1,0)$, providing a possibility of storing only one of them and obtaining the others by simple row/column exchanges and changes of the signs of certain elements. Certain rows of (6.20), like for example those corresponding to simulation of sinusoidal grid voltages, can be separately stored as sinusoidal functions. Also, usage of a plant model in $\alpha\beta$ frame instead of in abc reduces the memory requirement by making the size of the involved matrix exponentials smaller.

6.5.2 Constraint Set Approximation of the QP

The QP approximation (6.60) can be addressed by a large number of optimization algorithms such as interior-point, active-set or operator splitting methods (see, e.g., [37, 53]). To efficiently obtain an approximate solution of the QP (6.60) by a simple gradient-projection algorithm, an additional approximation will be introduced by removing the sequence constraint set Ξ_s and leaving only the box constraint Ξ_b. The main motivation for this simplification is the projection substep involved in each gradient projection iteration whose execution can be performed with particular computational efficiency in case of a box constraint shape.

The removing of the sequence constraint is additionally justified by the effect which the box constraint has, which is the limiting of the departure that the switching times can have from nominal switching times. This limiting reduces the probability that a change of the overall three-phase sequence can happen, making thus the impact of the removal of the sequence constraint less significant. In cases where this effect of the box constraint does not take place, a preventing of the sequence changes can be achieved during forming of the QP by adapting the upper and lover bounds in the box constraint in order to prohibit any change of sequence. In the numerical experiments of this chapter involving the considered case study, a necessity for utilization of an adaptation technique of this kind did not appear.

6.5.3 QP Solving: Early-Terminated Gradient-Projection

The obtained QP problem can be conveniently addressed by the gradient-projection optimization algorithm [13]. For notational convenience in what follows, the QP (6.60) is written as

$$\begin{aligned} &\underset{\Delta t_I}{\text{minimize}} \quad J_{QP}(\Delta t_I) \\ &\text{subject to} \quad \Delta t_I \in \Xi_b(\delta_{max}), \end{aligned} \tag{6.64}$$

where $J_{QP}(\Delta t_I)$ is the quadratic cost from (6.60) and $\Xi_b(\delta_{max})$ is the box constraint defined previously. Starting from an initial guess for the QP solution Δt^0, the gradient-projection algorithm with backtracking line search has the form

$$\Delta t_I^{p+1} = P_{\Xi_b}\left(\Delta t_I^p - \alpha^p \nabla J_{QP}(\Delta t_I^p)\right), \tag{6.65}$$

where $p = 0, 1, 2, \ldots$ is the iteration number, $P_{\Xi_b} : \mathbb{R}^{N_{sw}} \to \mathbb{R}^{N_{sw}}$ denotes projection on the set $\Xi_b(\delta_{max})$, and α^p is the stepsize at iteration p selected to be $\alpha^p = \beta^m \alpha_{gp}$ where α_{gp} is some user-specified initial stepsize for backtracking, $\beta \in (0,1)$, and m is the smallest positive integer for which the inequality

$$J_{QP}(\Delta t_I^p) - J_{QP}(\Delta t_I^{p+1}) \geq \sigma \nabla J_{QP}(\Delta t_I^p)^T \left(\Delta t_I^p - \Delta t_I^{p+1}\right)$$

holds, given some specified $\sigma \in (0,1)$. A good choice for the initial point is $\Delta t^0 = 0_{N_{sw}\times 1}$, which will be used in the numerical experiments of this chapter. Thanks to the box shape of the constraint set $\Xi_b(\delta_{max})$, the projection substep in (6.65) consists of a simple clipping of the components of the vector to be projected.

An alternative to the gradient-projection method could be its accelerated version [52] which involves an extrapolation mechanism between the iterations and can be shown to have better iteration complexity. Nevertheless, the performance of the simple gradient-projection encountered in the numerical experiments was satisfactory, and a need to resort to the accelerated version did not occur.

6.5.4 Iterative Trajectory Improvement

The previously described process of performing forward simulation, computing state sensitivities and forming a QP to obtain Δt_I can be iteratively performed multiple times by the controller within the control period. This multiple forming and solving of the QP problems would lead to an iterative trajectory improvement.

Starting with some $\bar{t}^0$, for which a good initialization is the OPP $\bar{t}^*$, for a certain number of trajectory improvement iterations N_{tr} the method would first perform a forward simulation of the system dynamics using the times $\bar{t}^q$ obtained from the previous iterate:

$$\bar{x}_k(\bar{t}^q) = e^{\bar{A}_{\ell^*_{I(k-1)}}(t^*_{I(k)} - t^*_{I(k-1)})} \bar{x}_{k-1}(\bar{t}^q), \tag{6.66}$$

in which $\ell^*_I(\bar{t}^q)$ is determined by the $\bar{t}^q$, and it would also formulate the matrices of state sensitivities $\nabla \bar{x}_k(\bar{t}^q)$ as well as the $N_k(\bar{t}^q)$ matrices. These components, which all have a

possibility to be evaluated by using memory-stored matrix exponentials, give rise to the QP optimization problem performed by linearizing at the times $\bar{t}^q$:

$$\begin{aligned} &\underset{\Delta t_I}{\text{minimize}} \;\; \Delta t_I^T H(\bar{t}^q)\Delta t_I + h(\bar{t}^q)\Delta t_I + J_{\Delta ds}(\Delta t_I; \bar{t}^q) \\ &\text{subject to} \;\; \Delta t_I \in \Xi_b(\delta_{max}). \end{aligned} \tag{6.67}$$

The optimal solution of the above QP, the Δt_I^q, determines the modification $\Delta \bar{t}^q$ which should be used to obtain the next iterate $\bar{t}^{q+1} = \bar{t}^q + \Delta \bar{t}^q$. The process is repeated until N_{tr} iterations of trajectory improvement are executed.

The $J_{\Delta ds}(\Delta t_I; \bar{t}^q)$ quadratic penalty in (6.67) can be chosen as a penalty on the deviations from the nominal times $\bar{t}^*$:

$$J_{\Delta ds}(\Delta t_I; \bar{t}^*) = (t_I^q + \Delta t_I - t_I^*)^T (t_I^q + \Delta t_I - t_I^*), \tag{6.68}$$

in which the t_I^q is determined by $\bar{t}^q$, or as a penalty from the previous iterate $\bar{t}^q$:

$$J_{\Delta ds}(\Delta t_I; \bar{t}^q) = (\Delta t_I)^T (\Delta t_I), \tag{6.69}$$

which will be the form used in the numerical experiments.

As can be seen, the QP in (6.67) has the sequence constraint Ξ_s removed, which after iterating the above procedure for a certain number of iterations could cause a t_I^{q+1} whose components are not monotonically non-decreasing. This violation can be prevented at the QP forming phase by an adaptive box constraint tightening which would modify the box-constraint bounds so that a change of three-phase sequence (i.e., violation of Ξ_s) is not possible. Another possibility is to solve the QP without box-constraint adaptation and use some simple post-processing of the obtained switching times in t_I^{q+1} to correct its elements into a non-decreasing sequence. For instance, this could be achieved by modifying the components t_{Ii}^{q+1} which are breaking the sequence constraint (i.e., components for which $t_{I(i-1)}^{q+1} > t_{Ii}^{q+1}$) to be equal to their preceding $t_{I(i-1)}^{q+1}$. This simple post-processing approach was utilized in the numerical experiments of this chapter.

It should be observed that the effect of the described iterative trajectory improvement may also be introduced by solving only one QP per control period and passing its solution to the next control period for forming the new QP. This can be observed as an iterative trajectory improvement where after each solving of the QP, the initial portion of the obtained input trajectory is applied to the system and the remaining part of the obtained trajectory is then further processed by the QP in the next control period.

6.6 Performance Evaluation

The described control method has been tested in simulation by using Matlab. The performances of the approximate controller versions was benchmarked against the performance of the original unapproximated OCP based on OPPs (6.31) which was numerically tackled by applying the gradient method as described in Section 6.4.4. The parameters of the medium-voltage grid-tied power converter with *LC* filter are taken from [69] and are listed in Table 6.2. The control period used in the experiments is set to $T_s = 50\mu s$, the pulse number of the used OPP is $d = 8$ and the number of switches within the prediction horizon is $N_{sw} = 9$. The matrix exponentials were stored for the range $[0, \tau_{max}]$ with $\tau_{max} = 2$ ms and $n_g = 10$ gridding points, and they were used online by doing convex combinations of them. A summary of the controller parameters used in the experiments is given in Table 6.3.

The results in figures are given with vertical axes normalized with respect to the base quantities. The selected base values for voltage, current, power and frequency are $V_B = \sqrt{2/3}V_g$, $I_B = \sqrt{2}I_{gn}$, $S_B = 8$ MVA and $f_B = 50$ Hz.

The QP approximation of the controller has been implemented on an FPGA model Kintex KCU 1500, and the C code was synthesized using Vivado HLS. Table 6.1 summarizes the latency (the number of clock cycles required for execution) and usage of FPGA resources for an implementation optimized using pipelining and loop unrolling. Due to clock inaccuracy of $1.25ns$, the worst clock time is $10ns$, resulting thus in $60\mu s$ required for execution of the control method on the available FPGA model. The implementation on the available FPGA demonstrates that an execution of the control method can be performed with time which is within the order of magnitude of the desired one. Further decrease of the computation time could be achieved by introducing more paralelism into the implementation, which would require an FPGA with an additional amount of resources. The FPGA implementation is not a contribution of this thesis, and its description is thus not further detailed.

Method	QP formulation	QP solve	total
latency	5956	66	6022
clk $[ns]$	8.74	8.74	8.74
BRAM (%)	41	0	41
DSP (%)	92	7	99
FF (%)	19	1	20
LUT (%)	96	4	100

Table 6.1 – Latency and resource consumption on Kintex KCU 1500

Table 6.2 – Converter, *LC* filter and grid parameters.

Parameter		*Value*
Converter's DC link voltage	v_{dc}	5200 V
LCL inverter-side inductance	L_{fi}	600 μH
LCL inverter-side resistance	R_{fi}	5 mΩ
LCL capacitance	C_{f}	1 mF
LCL grid-side inductance	L_{fg}	600 μH
LCL grid-side resistance	R_{fg}	5 mΩ
Grid voltage	U_{g}	3000 V
Grid nominal current	I_{gn}	1540 A
Grid frequency	f_{g}	50 Hz

6.6.1 Transient Performance

The transient performance of the controller has been tested by introducing step changes of the reference active-reactive power pair (P_g, Q_g). In particular, the tests were done by keeping the reactive power reference at $Q_g = 0$ and changing the reference active power P_g. The reactive power reference $Q_g = 0$ is a typical choice in the grid-connected power electronics applications since it results in the best power factor. Results obtained when changing the power reference from $(0.6, 0)$ in per-unit to $(0, 0)$ are represented in Fig. 6.6. It can be seen in the figure that an increase of the number of QP iterations executed per control period leads to a closer approximation of the performance obtained with the original unapproximated OCP based on OPPs. The difference between the OCP and the approximate versions is less pronounced for smaller step changes of references, as can be seen for example in Fig. 6.7 where a change of reference from $(0.8, 0)$ to $(0.4, 0)$ was involved.

Transient performance is also illustrated in Fig. 6.8 where the total (three-phase) instantaneous power injected into the grid is plotted for the two cases illustrated in Fig. 6.6 and Fig. 6.7. It can be observed how an increased number of QP iterations leads to a performance closer to the unapproximated OCP. It should be noted that although the instantaneous power of each single phase is not constant, the total three-phase instantaneous power is constant in steady-state, as depicted in Fig. 6.8. Plots of instantaneous reactive power injected into the grid are not considered as instantaneous reactive power is not a defined concept in electric circuits theory.

Table 6.3 – Controller parameters.

Parameter		*Value*
Control period	T_s	$50\,\mu$s
OPP pulse number	d	8
Switches in prediction horizon	N_sw	9
Lower bound for the modulation indices	m_{min}	0
Upper bound for the modulation indices	m_{max}	$4/\pi$
Number of modulation index gridding points	n_m	256
Upper bound for storing of exponentials	τ_{max}	2 ms
Number of gridding points for exponentials	n_g	10
Penalties in Q for inverter current	Q_fi	0
Penalties in Q for grid current	Q_fg	1
Penalties in Q for capacitor voltage	Q_f	0.1
Number of gradient-projection iterations for QP	N_gp	10
QP's bound of box constraint	δ_{max}	$30\,\mu$s

6.6.2 Steady-State Performance

The steady-state performance of the controller is quantified by total harmonic distortion (THD) of the grid current, which is for the current in each phase $p \in \{a,b,c\}$ defined as:

$$\mathrm{THD}_{\%p} = \frac{\sqrt{\sum_{k=2}^{\infty} \hat{i}_{\mathrm{fg}pk}^2}}{\hat{i}_{\mathrm{fg}p1}} \cdot 100\%, \tag{6.70}$$

with $\hat{i}_{\mathrm{fg}pk}$ denoting the k-th harmonic of the given phase signal $i_{\mathrm{fg}p}(t)$. An average THD over the three phases is given by

$$\mathrm{THD}_{\%} = \frac{1}{3}\left(\mathrm{THD}_{\%a} + \mathrm{THD}_{\%b} + \mathrm{THD}_{\%c}\right). \tag{6.71}$$

The THD values of different controller variants obtained at $(P_g, Q_g) = (0.8, 0)$ with harmonics computed by using one steady-state period are given in Table 6.4. It can be seen that the approximate versions involve a slight degradation of the steady-state performance in comparison to the unapproximated OCP solution, although the values obtained are still very low and demonstrate the performance of steady-state operation with combined usage of OPPs and *LC* filter.

6.7 Conclusions

This chapter described an optimization-based power electronics control method based on OPPs. The method is developed by first introducing an OCP based on OPPs which is then

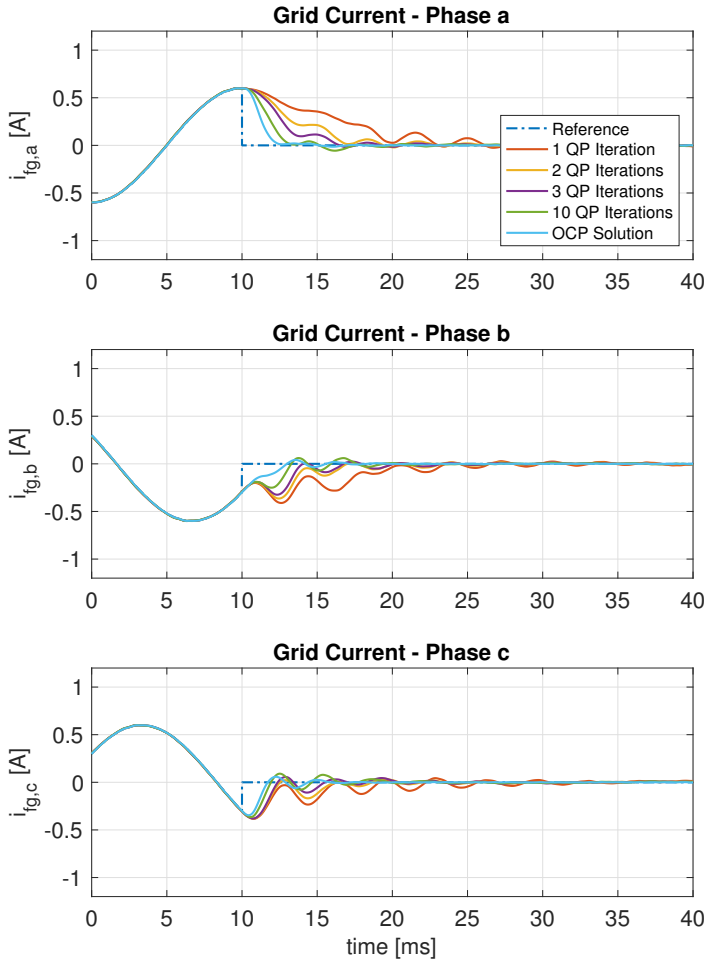


Figure 6.6 – A step change of the power reference pair (P_g, Q_g) at 10 ms from $(0.6, 0)$ to $(0, 0)$.

addressed in order to obtain low-computational complexity approximations of its control policy. In comparison to other methods which use OPPs to store the optimal steady-state operation in controller's memory, the method of this chapter uses integrated squared tracking error as a cost function and also allows a usage of memory to store the information about dynamic system behavior, thereby allowing an approximate low-computational complexity MPC that optimizes transient behavior of complex power electronics configurations such as a grid-tied power converter with *LC* filter. The obtained approximations provide a possibility for implementation of one of the controller's approximate versions on an FPGA.

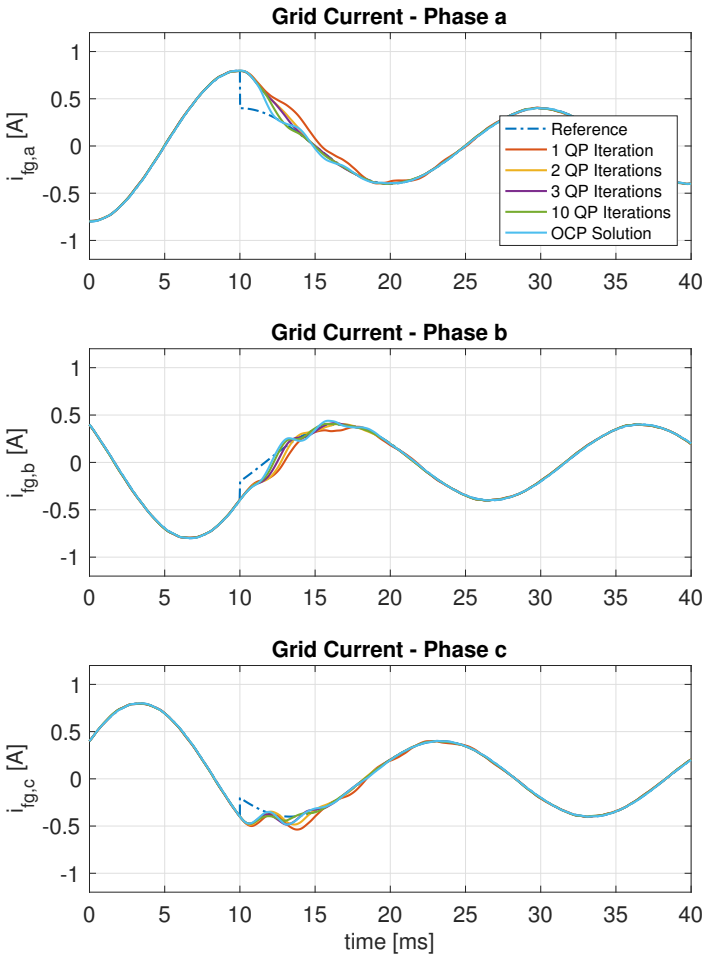
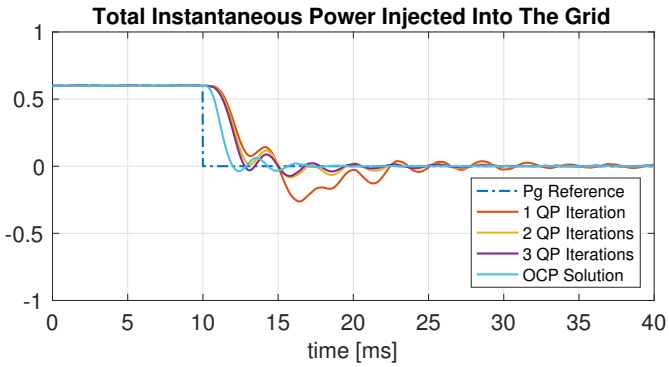


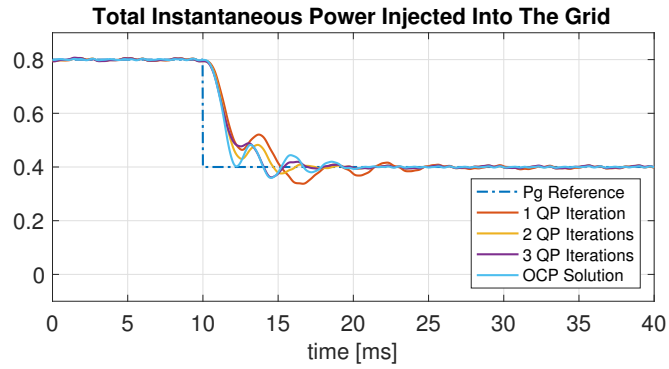
Figure 6.7 – A step change of the power reference pair (P_g, Q_g) at 10 ms from $(0.8, 0)$ to $(0.4, 0)$.

Table 6.4 – The THD values of different controller versions obtained at reference $(P_g, Q_g) = (0.8, 0)$.

Controller Version	THD$_{\%a}$	THD$_{\%b}$	THD$_{\%c}$	THD$_{\%}$
1 QP Iteration	0.48	0.42	0.48	0.46
2 QP Iterations	0.51	0.42	0.50	0.48
3 QP Iterations	0.54	0.50	0.56	0.53
10 QP Iterations	0.51	0.48	0.53	0.51
OCP Solution	0.26	0.26	0.26	0.26



(a) A change (P_g, Q_g) reference from $(0.6, 0)$ to $(0, 0)$.



(b) A change of (P_g, Q_g) reference from $(0.8, 0)$ to $(0.4, 0)$.

Figure 6.8 – Total instantaneous power injected into the grid in case of a step change of reference at 10 ms.

7 Conclusions

This thesis presented several contributions related to the computational complexity required for optimization-based controller execution. The main outcomes are summarized in what follows, as well as some possible directions for future research.

Part I - Automatic Controller Tuning

This part provided an automatic controller tuning method for constrained control of nonlinear systems. The method is capable of performing an offline controller tuning that optimizes some user-specified performance metric while imposing closed-loop stability. As such, it allows an easier design of low-complexity optimization-based controllers by performing tuning of their parameters (possibly also of the parameters in the optimization algorithm). The flexibility of the method also allows its application to tuning of the control structures that are usually applied in a heuristic manner, as well as application to novel controller forms that a user may find suitable for his problem (e.g., a control structure involving a user's intuitively designed optimization algorithm). Extensions of the initial sum-of-squares based controller tuning to multimodel uncertainty case and non-polynomial larger size systems are presented. The tuning procedure involves two phases, both of which are formulated as optimization problems that can be tackled by applying Bayesian optimization. The application examples involve synthesis of an explicit-MPC controller for speed control of the nonlinear PMSM model, an anti-windup equipped PID for gyroscope position control which is synthesized robust to multimodel uncertainty, an early terminated optimization-based controller for soft-constrained speed control of PMSM, and a cascaded linear controller with saturation for gyroscope position control.

A possible direction for future research is improvement of the method's scalability, both in terms of the number of controller tuning parameters that can be addressed as well as in terms of the size of the system (the number of states). Another possible direction is to extend the method into the safe reinforcement learning framework so that the performance metric is evaluated experimentally during tuning while the constraint satisfaction is enforced in

experiments to ensure safety until a controller with closed-loop stability and satisfactory performance is eventually obtained. Applications of the scenario-based tuning by using the control inputs directly computed by the available embedded hardware would also be an attractive direction, as the tuning method would be able to take into account the present round-off errors and tune as well the termination criteria of the optimization subalgorithms (e.g., termination criteria of a method solving subproblems of augmented Lagrangian algorithm) so that the overall early-terminated control policy computed by the embedded hardware is synthesized.

Part II - Accelerated ADMM based on Accelerated Douglas-Rachford Splitting

An accelerated version of ADMM based on accelerated Douglas-Rachford splitting is derived resulting in a method characterised by a $O(1/k^2)$ complexity of the internal convergence mechanism. In comparison to classical ADMM, the derived algorithm involves an additional algebraic step which corresponds to the extrapolations in the underlying fast gradient method. The numerical results show that the method can improve the performance of classical ADMM over the allowed range of penalty parameters, and that a heuristic modification can potentially extend the benefits of acceleration beyond this allowed range.

Besides exploration for more relaxed conditions which lead to guaranteed convergence properties, future work can also be oriented to the establishing of restarting schemes which can enforce convergence of the heuristic accelerated version. The heuristic version's acceleration also can be exploited by the tuning methods of part one of the thesis, which can arrange the penalty parameter and the overrelaxation factor such that the closed-loop system behavior is directly addressed.

Part III - Power Converter Pulse Patterns Optimal Control

This part introduced an optimization-based power electronics control method using OPPs. In comparison to the other methods which use OPPs to store the optimal steady-state operation in the controller's memory, the method of this chapter uses integrated squared tracking error as the controller's cost function and also allows a usage of memory to store the information about the system's dynamic behavior. This allows an approximate low-computational complexity MPC that optimizes transient behavior of complex power electronics configurations, such as the grid-tied converter with *LC* filter which was considered as a case study. The low-complexity controller versions provide a possibility for implementation on embedded hardware. The approximate controller versions were tested in simulation and compared with the unapproximated optimal control based on OPPs in order to asses the impact which the approximations introduced for computational reasons have on the obtained performance.

Besides exploring possibilities for further computational reduction of the control method, future research can consider an experimental application on a power electronics test setup.

By using the method's flexibility in terms of the power electronics configurations that can be addressed, a testing of the method's performance when adapted to various problems of practical interest, such as for example the case of non-symmetric operating conditions, is also a valuable direction for further investigation.

Bibliography

[1] Amir Ali Ahmadi and Georgina Hall. Dc decomposition of nonconvex polynomials with algebraic techniques. *arxiv.org*, 2015. URL https://arxiv.org/abs/1510.01518.

[2] Mohamadreza Ahmadi, Hamed Mojallali, and Rafael Wisniewski. Guaranteed cost h-infinity controller synthesis for switched systems defined on semi-algebraic sets. *Nonlinear Analysis: Hybrid Systems*, 11:37 – 56, 2014. ISSN 1751-570X. doi: https://doi.org/10.1016/j.nahs.2013.04.001.

[3] S. Almér. Model Predictive PWM of a Single Phase Inverter: A Nonlinear Transformation Approach. In *Energy Conversion Congress and Exposition (ECCE), 2015 IEEE*, pages 4301–4306, Sept 2015. doi: 10.1109/ECCE.2015.7310268.

[4] Marc Teboulle Amir Beck. Gradient-based algorithms with applications to signal recovery problems. *Convex Optimization in Signal Processing and Communications*, 2009.

[5] MOSEK ApS. *The MOSEK optimization toolbox for MATLAB manual. Version 8.1.*, 2017. URL http://docs.mosek.com/8.1/toolbox/index.html.

[6] H.H. Bauschke. *Projection Algorithms and Monotone Operators*. Canadian theses. Thesis (Ph.D.)–Simon Fraser University, 1996. ISBN 9780612167896.

[7] Dimitri Bertsekas. *Dynamic Programming and Optimal Control: Volume 1*. Athena Scientific, 2005.

[8] Dimitri Bertsekas. *Dynamic Programming and Optimal Control: Volume 2*. Athena Scientific, 2006.

[9] Dimitri Bertsekas and John Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice hall, 1989.

[10] Dimitri Bertsekas and John Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[11] Dimitri P. Bertsekas. *Nonlinear Programming: 2nd Edition*. Athena Scientific, 2004.

[12] Dimitri P. Bertsekas. *Convex Optimization Theory*. Athena Scientific, 2009.

[13] Dimitri P. Bertsekas. *Convex Optimization Algorithms*. Athena Scientific, 2015.

[14] F. Blaschke. The principle of field orientation applied to the new transvector closed-loop control system for rotating field machines. *Siemens Rev.*, 39:217–220, 1972.

[15] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521833787.

[16] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2010. ISSN 1935-8237.

[17] Eric Brochu, Vlad M. Cora, and Nando de Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *CoRR*, abs/1012.2599, 2010. URL http://arxiv.org/abs/1012.2599.

[18] G. S. Buja. Optimum output waveforms in pwm inverters. *IEEE Transactions on Industry Applications*, IA-16(6):830–836, Nov 1980. ISSN 0093-9994. doi: 10.1109/TIA.1980.4503879.

[19] G. C. Calafiore and M. C. Campi. The scenario approach to robust control design. *IEEE Transactions on Automatic Control*, 51(5):742–753, May 2006. ISSN 0018-9286. doi: 10.1109/TAC.2006.875041.

[20] Giuseppe Carlo Calafiore. Random convex programs. *SIAM Journal on Optimization*, 20 (6):3427–3464, 2010. doi: 10.1137/090773490. URL https://doi.org/10.1137/090773490.

[21] M. C. Campi and S. Garatti. The exact feasibility of randomized solutions of uncertain convex programs. *SIAM Journal on Optimization*, 19(3):1211–1230, 2008. doi: 10.1137/07069821X. URL https://doi.org/10.1137/07069821X.

[22] R. H. Cannon. *Dynamics of physical systems*. Courier Dover Publications, 2003.

[23] N. Celanovic and D. Boroyevich. A fast space-vector modulation algorithm for multilevel three-phase converters. *IEEE Transactions on Industry Applications*, 37(2):637–641, March 2001. ISSN 0093-9994. doi: 10.1109/28.913731.

[24] G. Cimini, D. Bernardini, A. Bemporad, and S. Levijoki. Online model predictive torque control for permanent magnet synchronous motors. In *2015 IEEE International Conference on Industrial Technology (ICIT)*, pages 2308–2313, March 2015. doi: 10.1109/ICIT.2015.7125438.

[25] Edith Clarke. *Circuit analysis of AC power systems*, volume 1. Wiley, 1943.

[26] P. A. Dahono. A method to damp oscillations on the input lc filter of current-type ac-dc pwm converters by using a virtual resistor. In *The 25th International Telecommunications Energy Conference, 2003. INTELEC '03.*, pages 757–761, Oct 2003.

[27] J. Dannehl, F. W. Fuchs, S. Hansen, and P. B. Thogersen. Investigation of active damping approaches for pi-based current control of grid-connected pulse width modulation converters with lcl filters. *IEEE Transactions on Industry Applications*, 46(4):1509–1517, July 2010. ISSN 0093-9994. doi: 10.1109/TIA.2010.2049974.

[28] Marc Peter Deisenroth, Gerhard Neumann, and Jan Peters. A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142, 2013. ISSN 1935-8253. doi: 10.1561/2300000021. URL http://dx.doi.org/10.1561/2300000021.

[29] Jonathan Eckstein. Augmented lagrangian and alternating direction methods for convex optimization: A tutorial and some illustrative computational results. *RUTCOR Research Reports*, 2012.

[30] Jonathan Eckstein and DimitriP. Bertsekas. On the douglas-rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(1-3):293–318, 1992.

[31] T. Geyer. Generalized model predictive direct torque control: Long prediction horizons and minimization of switching losses. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pages 6799–6804, Dec 2009. doi: 10.1109/CDC.2009.5399829.

[32] T. Geyer. A comparison of control and modulation schemes for medium-voltage drives: Emerging predictive control concepts versus field oriented control. In *2010 IEEE Energy Conversion Congress and Exposition*, pages 2836–2843, Sep. 2010. doi: 10.1109/ECCE.2010.5618172.

[33] T. Geyer. Model predictive direct current control: Formulation of the stator current bounds and the concept of the switching horizon. *IEEE Industry Applications Magazine*, 18(2):47–59, March 2012. ISSN 1077-2618. doi: 10.1109/MIAS.2011.2175518.

[34] T. Geyer. *Model Predictive Control of High Power Converters and Industrial Drives*. Wiley, 2016. ISBN 9781119010906.

[35] T. Geyer, G. Papafotiou, and M. Morari. Model predictive direct torque control-part i: Concept, algorithm, and analysis. *IEEE Transactions on Industrial Electronics*, 56(6): 1894–1905, June 2009. ISSN 0278-0046. doi: 10.1109/TIE.2008.2007030.

[36] T. Geyer, N. Oikonomou, G. Papafotiou, and F. D. Kieferndorf. Model predictive pulse pattern control. *IEEE Transactions on Industry Applications*, 48(2):663–676, March 2012. ISSN 0093-9994. doi: 10.1109/TIA.2011.2181289.

[37] Tom Goldstein, Brendan O'Donoghue, Simon Setzer, and Richard Baraniuk. Fast alternating direction optimization methods. *SIAM Journal on Imaging Sciences*, 7(3):1588–1623, 2014. doi: 10.1137/120896219.

[38] K. Hasse. Zum dynamischen verhalten der asynchronmaschine bei betrieb mit variabler standerfrequenz und standerspannung. *ETZ-A Bd.*, 89:77, 1968.

[39] P. Hokayem, T. Geyer, and N. Oikonomou. Active damping for model predictive pulse pattern control. In *2014 IEEE Energy Conversion Congress and Exposition (ECCE)*, pages 1220–1227, Sep. 2014. doi: 10.1109/ECCE.2014.6953540.

[40] Grahame Holmes, Thomas Lipo, and T A Lipo. *Pulse Width Modulation for Power Converters: Principles and Practice*. John Wiley and Sons, 2003.

[41] H.K. Khalil. *Nonlinear Systems*. Pearson Education. Prentice Hall, 2002. ISBN 9780130673893.

[42] M. Korda and C. N. Jones. Certification of fixed computation time first-order optimization-based controllers for a class of nonlinear dynamical systems. In *2014 American Control Conference*, pages 3602–3608, June 2014. doi: 10.1109/ACC.2014.6858719.

[43] Milan Korda and Colin N. Jones. Stability and performance verification of optimization-based controllers. *Automatica*, 78:34 – 45, 2017. ISSN 0005-1098. doi: http://dx.doi.org/10.1016/j.automatica.2016.12.008.

[44] P.C. Krause, O. Wasynczuk, S.D. Sudhoff, and IEEE Power Engineering Society. *Analysis of electric machinery and drive systems*. IEEE Press series on power engineering. IEEE Press, 2002. ISBN 9780471143260.

[45] J. B. Lasserre. *Moments, Positive Polynomials and Their Applications*. Imperial College Press, first edition, 2009.

[46] Daniel Lizotte, Tao Wang, Michael Bowling, and Dale Schuurmans. Automatic gait optimization with gaussian process regression. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 944–949, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[47] C. Van Loan. Computing integrals involving the matrix exponential. *IEEE Transactions on Automatic Control*, 23(3):395–404, Jun 1978. ISSN 0018-9286. doi: 10.1109/TAC.1978.1101743.

[48] J. Löfberg. Yalmip : A toolbox for modeling and optimization in matlab. In *In Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

[49] MathWorks. Statistics and machine learning toolbox: User's guide (r2016b). *https://www.mathworks.com/help/pdf_doc/stats/stats.pdf*, September 2016.

[50] J. Meili, S. Ponnaluri, L. Serpa, P. K. Steimer, and J. W. Kolar. Optimized pulse patterns for the 5-level anpc converter for high speed high power applications. In *IECON 2006 - 32nd Annual Conference on IEEE Industrial Electronics*, pages 2587–2592, Nov 2006. doi: 10.1109/IECON.2006.347555.

[51] A. Nabae, I. Takahashi, and H. Akagi. A new neutral-point-clamped pwm inverter. *IEEE Transactions on Industry Applications*, IA-17(5):518–523, Sep. 1981. ISSN 0093-9994. doi: 10.1109/TIA.1981.4503992.

[52] Yurii Nesterov. *Introductory lectures on convex optimization - A basic course*. Springer, 2004.

[53] Jorge Nocedal and Stephen J Wright. *Numerical Optimization, Second Edition*. Springer New York, 2006.

[54] N. Oikonomou, C. Gutscher, P. Karamanakos, F. Kieferndorf, and T. Geyer. Model predictive pulse pattern control for the five-level active neutral point clamped inverter. In *Energy Conversion Congress and Exposition (ECCE), 2012 IEEE*, pages 129–136, Sept 2012. doi: 10.1109/ECCE.2012.6342832.

[55] G. Papafotiou, J. Kley, K. G. Papadopoulos, P. Bohren, and M. Morari. Model predictive direct torque control-part ii: Implementation and experimental evaluation. *IEEE Transactions on Industrial Electronics*, 56(6):1906–1915, June 2009. ISSN 0278-0046. doi: 10.1109/TIE.2008.2007032.

[56] Pablo A. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical Programming*, 96(2):293–320, 2003. ISSN 1436-4646.

[57] P. Patrinos, L. Stella, and A. Bemporad. Douglas-rachford splitting: Complexity estimates and accelerated variants. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 4234–4239, Dec 2014. doi: 10.1109/CDC.2014.7040049.

[58] I. Pejcic and C. N. Jones. Accelerated admm based on accelerated douglas-rachford splitting. In *2016 European Control Conference (ECC)*, pages 1952–1957, June 2016. doi: 10.1109/ECC.2016.7810577.

[59] I. Pejcic, S. Almér, and H. Peyrl. Voltage source converter mpc with optimized pulse patterns and minimization of integrated squared tracking error. In *2017 American Control Conference (ACC)*, pages 4069–4074, May 2017. doi: 10.23919/ACC.2017.7963579.

[60] Ivan Pejcic and Colin N. Jones. Experimental verification of sum-of-squares-based controller tuning technique with extension to parallel multimodel uncertainty processing. In *European Control Conference*, 2019.

[61] Ivan Pejcic, Milan Korda, and Colin N. Jones. Control of nonlinear systems with explicit-mpc-like controllers. In *The 56th IEEE Conference on Decision and Control*, 2017.

[62] Athanasios S. Polydoros and Lazaros Nalpantidis. Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, 86(2):153–173, May 2017. ISSN 1573-0409. doi: 10.1007/s10846-017-0468-y. URL https://doi.org/10.1007/s10846-017-0468-y.

[63] S. Prajna, A. Papachristodoulou, and Fen Wu. Nonlinear control synthesis by sum of squares optimization: a lyapunov-based approach. In *2004 5th Asian Control Conference (IEEE Cat. No.04EX904)*, volume 1, pages 157–165 Vol.1, July 2004.

[64] S. Prajna, P. A. Parrilo, and A. Rantzer. Nonlinear control synthesis by convex optimization. *IEEE Transactions on Automatic Control*, 49(2):310–314, Feb 2004. ISSN 0018-9286. doi: 10.1109/TAC.2003.823000.

[65] James A. Primbs. Brief the analysis of optimization based controllers. *Automatica*, 37(6): 933–938, June 2001. ISSN 0005-1098.

[66] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006. ISBN 026218253X.

[67] A. K. Rathore, J. Holtz, and T. Boller. Synchronous optimal pulsewidth modulation for low-switching-frequency control of medium-voltage multilevel inverters. *IEEE Transactions on Industrial Electronics*, 57(7):2374–2381, July 2010. ISSN 0278-0046. doi: 10.1109/TIE.2010.2047824.

[68] James Blake Rawlings and David Q Mayne. *Model predictive control: Theory and design*. Nob Hill Pub., 2009.

[69] J. Scoltock, T. Geyer, and U. Madawala. Model predictive direct power control for a grid-connected converter with an lcl-filter. In *Industrial Technology (ICIT), 2013 IEEE International Conference on*, pages 588–593, Feb 2013. doi: 10.1109/ICIT.2013.6505737.

[70] Daniel Simon and Johan Löfberg. Stability analysis of model predictive controllers using mixed integer linear programming. *arxiv.org*, 2016. URL https://arxiv.org/abs/1604.00863.

[71] I. Takahashi and T. Noguchi. A new quick-response and high-efficiency control strategy of an induction motor. *IEEE Transactions on Industry Applications*, IA-22(5):820–827, Sep. 1986. ISSN 0093-9994. doi: 10.1109/TIA.1986.4504799.

[72] G. Valmorbida, S. Tarbouriech, and G. Garcia. State feedback design for input-saturating nonlinear quadratic systems. In *2009 American Control Conference*, pages 1231–1236, June 2009. doi: 10.1109/ACC.2009.5160428.

[73] Lieven Vandenberghe. Optimization methods for large-scale systems, course slides. *University of California, Los Angeles*, Spring 2013-14.

[74] Aaron Wilson, Alan Fern, and Prasad Tadepalli. Using trajectory data to improve bayesian optimization for reinforcement learning. *Journal of Machine Learning Research*, 15: 253–282, 2014. URL http://jmlr.org/papers/v15/wilson14a.html.

**Ce document a été imprimé au Centre d'impression EPFL,
imprimerie climatiquement neutre, certifiée myClimate.**