SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF CHEMICAL AND FOOD TECHNOLOGY

Reference number: FCHPT-19990-50911

FAST AND MEMORY-EFFICIENT IMPLEMENTATION OF MODEL PREDICTIVE CONTROL

DISSERTATION THESIS

Bratislava, 2016 **Ing. Juraj Holaza**

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF CHEMICAL AND FOOD TECHNOLOGY

FAST AND MEMORY-EFFICIENT IMPLEMENTATION OF MODEL PREDICTIVE CONTROL

DISSERTATION THESIS

FCHPT-19990-50911

Study program: Process Control
Study field number: 2621
Study field: 5.2.14 Automation
Workplace: Department of Information Engineering and Process Control
Supervisor: doc. Ing. Michal Kvasnica, PhD.

Bratislava, 2016 **Ing. Juraj Holaza**

Slovak University of Technology in Bratislava

Institute of Information Engineering, Automation and Mathematics

Faculty of Chemical and Food Technology

DISSERTATION THESIS TOPIC

Author of thesis:	Ing. Juraj Holaza
Study programme:	Process Control
Study field:	5.2.14. automation
Registration number:	FCHPT-19990-50911
Student's ID:	50911
Thesis supervisor:	doc. Ing. Michal Kvasnica, PhD.
Title of the thesis:	**Fast and Memory-Efficient Implementation of Model Predictive Control**
Date of entry:	01. 09. 2012
Date of submission:	31. 05. 2016

Ing. Juraj Holaza
solver

prof. Ing. Miroslav Fikar, DrSc.
Head of department

prof. Ing. Miroslav Fikar, DrSc.
Study programme supervisor

Acknowledgements

In this opportunity, I would like to address my gratitude to all people who helped me to get where I am now. Firstly, I would like to thank to my supervisor associate professor Michal Kvasnica who has guided me for four years through my entire PhD study. Particularly, I am grateful for his constructive consultations and criticism, for giving me uncountable valuable advises and, most importantly, that he gave me research freedom during my study. My gratefulness also goes to professor Miroslav Fikar who has, beside of co-supervising me, created a skillful team at the Institute of Information Engineering, Automation, and Mathematics, with whom I have had a pleasure of working with. Here, I would especially thank to all professors Monika Bakošová, Ján Dvoran, Alajos Mészáros and Ján Mikleš for their help during my studies, to Katarína Macušková and Andrea Kalmárová for giving me a helping hand with all of the administration, to our technician Stanislav Vagač who has always found the right component that I needed, to my office mate Bálint Takács for creating a friendly environment and for his cooperation, and to all of my colleagues Slavomír Blažek, Ľuboš Čirka, Ján Drgoňa, Deepak Ingole, Martin Jelemenský, Martin Kalúz, Martin Klaučo, Juraj Oravec, Daniela Pakšiová, Ayush Sharma, Alexander Szücs, Juraj Števek, Richard Valo, and Anna Vasičkaninová. I am deeply grateful for spending time with these people who have created for me a second home.

I also can not forget to thank my colleagues in Zagreb. Especially, associate professor Mato Baotić for inviting me for research stay at Faculty of Electrical Engineering and Computing, University of Zagreb, Vinko Lešić for co-supervising me, and to my mates, Anita Martinčević, Antonio Starčić and Hrvoje Novak. I

have really enjoined our cooperation.

And last, but not least, my big thanks goes to my closest friends, who have cheered me and kept me sane during my study, and most importantly I am grateful to my family who has supported me during me entire life.

Juraj Holaza
Bratislava, 2016

Abstract

Model predictive control (MPC) represents a state-of-the-art control strategy which allows one to incorporate all of the system's constraints directly into the optimal control problem. The main limitation of MPC, however, is that complexity of such optimal control problems might exceed available resources of the implementation hardware. If exceeded, successful implementation of MPC is jeopardized and one needs to perform additional measures to tackle this issue. This thesis proposes a twofold answer to this issue and illustrates applicability of MPC to two practical case studies.

In the first direction, the concept of explicit MPC is exploited to mitigate online computational requirements of MPC. In order to keep the memory consumption of these explicit controllers on a tractable level, several complexity reduction techniques are introduced. Proposed methods aim at decreasing the complexity of explicit MPC via approximating the original (complex) solution by its simpler version. This approximation is performed in such a way that the approximation error between these two counterparts is minimized and recursive feasibility and asymptotic closed-loop stability is enforced.

The second direction suggests to decrease the complexity of MPC policy in a different manner. It is reported that certain system constraints might be omitted from the MPC formulation and then a-posteriori verified whether they are voluntarily enforced (e.g. by a proper tuning). Specifically, it is shown that multiparametric programming can be used as a tool to determine if the MPC feedback law exhibits properties like recursive feasibility or asymptotic closed-loop stability even when its control inputs are subjected to rounding-based quantization. Moreover, a safety property of MPC policies is verified, i.e. determining whether a set of unsafe states is avoided by the control strategy from a given set of initial conditions.

The last part of this thesis is devoted to application of MPC to real-life processes. Two plants are considered. In the first case study we aim at controlling pH of a chemical reaction between acetic acid and sodium hydroxide. The second process is a wind turbine, where the goal is to design a controller which performs well and provides safety operation.

Keywords: explicit model predictive control, multiparametric programming, reachability, quantization, pH control, wind turbine

Abstrakt

Prediktívne riadenie (MPC) predstavuje moderný prístup k riadeniu, ktorý dovoľuje zostrojiť regulátory pre mnohorozmerové systémy pri súčasnom zohľadnení všetkých fyzických a výkonnostných ohraničení, ktoré sú priamo zakomponované v optimalizačnom probléme. Avšak hlavnou prekážkou tejto stratégie je vysoká výpočtová náročnosť, ktorá často prevyšuje dostupné prostriedky bežných (nízkorozpočtových) riadiacich hardvérov. Predkladaná práca navrhuje dve principiálne možnosti, ako znížiť tieto požiadavky a súčasne poskytuje dve aplikácie tejto metodológie.

V prvom prístupe, za účelom zníženia výpočtových nárokov, sa využíva koncept explicitného MPC. Následne je predložená metóda, ktorá navrhuje ako zmierniť aj pamäťové nároky tohto prístupu, a to prostredníctvom aproximácie pôvodného explicitného riešenia jeho jednoduchšou verziou. Je preukázané, že uvedený prístup ma vždy aspoň jedno prípustné riešenie a dokáže efektívne znížiť zložitosť explicitných regulátorov pri súčasnom zachovaní rekurzívnej riešiteľnosti a zavedenia asymptotickej spätnoväzbovej stability. Avšak, cenou za takéto zníženie pamäťovej náročnosti je mierny pokles výkonnosti aproximovaného regulátora.

Druhý smer práce navrhuje odlišný prístup zníženia zložitosti MPC stratégie. Hlavnou myšlienkou je zachovanie konvexnosti optimalizačných problémov prostredníctvom vynechania určitých ohraničení z matematických formulácií a následnou verifikáciou identifikovať, či tieto riadiace vlastnosti sú aj napriek tomu implicitne splnené (napríklad dôkladným ladením regulátora). Konkrétne, táto práca predkladá rigorózne certifikáty, ktoré preverujú vlastnosti MPC zákonov riadenia, ktorých akčné zásahy sú podmienené kvantizačným pravidlom, ale aj bezpečnosť v zmysle vyhýbania sa nebezpečným stavom riadeného systému.

Praktická aplikácia MPC tvorí poslednú časť tejto práce. Prvým systém je chemický reaktor, kde hlavnou úlohou je riadiť pH chemickej reakcie medzi kyselinou octovou a hydroxidom sodným na požadovanú hodnotu. Druhým procesom je veterná turbína, kde cieľom je navrhnúť regulátor, ktorý ma dobré riadiace vlastnosti a zaručuje bezpečnú prevádzku turbíny.

Kľúčové slová: explicitné prediktívne riadenie, parametrické programovanie, dosiahnuteľnosť, kvantizácia, riadenie pH, veterná turbína

Contents

List of Figures

List of Tables

Abbreviations

DMC	Dynamic Matrix Control
ISE	Integral of Squared Error
KF	Kalman Filter
KKT	Karush-Kuhn-Tucker
FOC	Field Oriented Control
LICQ	Linear Independence Constraint Qualification
LO	Luenberger Observer
LP	Linear Program
LQR	Linear Quadratic Regulator
LTI	Linear Time-Invariant
MILP	Mixed Integer Linear Program
MIMO	Multiple-Input Multiple-Output
MIP	Mixed Integer Program
MIQP	Mixed Integer Quadratic Program
MPC	Model Predictive Control
MPHC	Model Predictive Heuristic Control
ODE	Ordinary Differential Equation
ORM	Optimal Region Merging
PID	Proportional Integral Derivative Controller
PL	Point Location
PLC	Programmable Logic Controller

PWA	Piecewise Affine
PWL	Piecewise Linear
PWQ	Piecewise Quadratic
QP	Quadratic Program
RHC	Receding Horizon Control
SISO	Single-Input Single-Output
TSP	Traveling Salesman Problem
mp-LP	Multiparametric Linear Programming
mp-QP	Multiparametric Quadratic Programming

Chapter 1

Introduction

" Either write something worth reading or do something worth writing. "

Benjamin Franklin

Model predictive control (MPC) represents an advanced control strategy which enjoys a wide popularity, especially in the industrial field. It is endorsed mainly due to its natural capability of designing feedback controllers for large multiple-input multiple-output (MIMO) systems, while enforcing all of the system's physical constraints which are explicitly embedded in the optimization problem. Moreover, MPC employs a mathematical model, which approximates dynamics of the controlled system, based on which it predicts the future evolution of the plant. This enables MPC to optimize control actions on the entire (finite) prediction horizon, hence increasing its control performance. Based on these advantages, MPC can reduce not only the material and energy consumption, but also minimize negative impacts on the environment, while maximizing the profit. These economic and environmental aspects are nowadays attracting an increased attention in the design of advanced control algorithms. On the other hand, the main limitation of this control approach lies in the computational demands, because in order to maintain guarantees of optimality, stability and constraint satisfaction, the whole optimization problem has to be solved within the duration of one sampling period. However, such a requirement can be easily jeopardized if not enough computational

resources are available in the control hardware, or if the sampling time decreases.

Therefore, at the beginning of this century a new method has been established which is called the explicit MPC. This approach tries to abolish the aforementioned computational limitations by means of offline pre-calculation of the whole optimization problem for all feasible initial conditions by using multi-parametric programming. The solution of such a computation is an explicit optimizer, encoded as a piecewise affine function (PWA) defined over a certain polytopic partition, which maps measurements of states onto the optimal control actions. Subsequently, after implementation of this optimizer into control hardware, the whole online procedure of obtaining optimal control inputs is then reduced only to a mere function evaluation, which can be performed efficiently even with a small computation power. And this is the reason why explicit MPC opens a new possibilities of employing MPC policy even to processes with fast dynamics, where the traditional implicit MPC implementation was impracticable due to rapid sampling rates or software reliability issues.

Fast evaluation of optimal control inputs is not the only feature of explicit MPC. With the parametric solution in hand one has also better understand the control behavior and properties of MPC, compared to its equivalent implicit formulation. This possibility becomes very handy for the purposes of control analysis, where one can validate properties such as closed-loop stability, optimality, robustness, and many others, all of which play a crucial role in control theory.

Complexity Reduction in Explicit MPC

Even though explicit MPC is an attractive control strategy when aiming at fast implementation of optimization-based controllers with limited control resources, it also faces several theoretical and technical limitations. The main obstacle is a high memory consumption of explicit solutions, which limits their practical implementation. It is important to point out that while the complexity of those optimizers, in a terms of number of regions over which they are defined, can grow exponentially with the prediction horizon, the commonly used industrial hardware (such as programmable logic controllers or embedded microchips) have their available memory strictly limited only up to several kilobytes. Therefore, it is of paramount importance to keep the complexity of explicit MPC controllers at an acceptable level. The evident advantages coming out of explicit MPC strategy have attracted

a lot of interested researchers, which have put a lot of effort into development of techniques that try to mitigate the implied disadvantages. One of the directions of the research community is aimed towards reducing the complexity of explicit MPC controllers in order to secure a successful implementation on the control hardware. This direction is also referred to as a memory reduction in explicit MPC.

In this thesis we elaborate possibilities of abolishing the aforementioned hardware limitations. Particularly, we propose a novel method of reducing complexity of explicit MPC solutions, which belongs to the class of methods which trade lower complexity for a certain reduction of performance. In the presented method, however, the loss of performance is mitigated as much as possible, hence achieving nearly-optimal performance with low complexity. The method is based on the assumption that a complex explicit MPC feedback law $\mu(x)$ is given, encoded as a PWA function of the state measurements x. Our objective is to replace $\mu(\cdot)$ by a simpler PWA function $\widetilde{u}(\cdot)$ such that: (i) $\widetilde{u}(x)$ generates a feasible sequence of control inputs for all admissible values of x; (ii) $\widetilde{u}(x)$ renders the closed-loop system asymptotically stable; and (iii) the integrated squared error between $\mu(\cdot)$ and $\widetilde{u}(\cdot)$ (i.e., the suboptimality of $\widetilde{u}(\cdot)$ with respect to $\mu(\cdot)$) is minimized. By doing so we obtain a simpler explicit feedback law $\widetilde{u}(\cdot)$, which is safe (i.e., it provides constraint satisfaction and closed-loop stability), and is nearly optimal.

Designing an appropriate approximate controller $\widetilde{u}(\cdot)$ requires first the construction of the polytopic regions over which $\widetilde{u}(\cdot)$ is defined, and then the synthesis of local affine expressions in each of the regions. We propose to approach the first task by solving a simpler MPC optimization problem with a shorter prediction horizon. In this way, we obtain a simple feedback $\widehat{\mu}(\cdot)$ as a PWA function. However, such a simpler feedback typically exhibits large deterioration of performance compared to $\mu(\cdot)$. To mitigate such a performance loss, we retain the regions of $\widehat{\mu}(\cdot)$, but refine the associated local affine feedback laws to obtain the function $\widetilde{u}(\cdot)$ such that the error between $\mu(\cdot)$ and $\widetilde{u}(\cdot)$ is minimized.

Performance Verification of MPC

Two directions of performance verification are followed in this thesis. Both of them are essentially trying to reduce the complexity of MPC strategy such that implementation on commonly used industrial hardware will be enabled. In the first approach we define rigorous conditions under which a given explicit MPC

controller provides guarantees of closed-loop stability and recursive constraint satisfaction when its control commands are quantized, e.g. by a D/A converter. This direction is motivated by employing explicit MPC strategies in control of systems with quantized inputs, such as elevators. The second approach aims at verifying safety of MPC policies. Here, we provide a rigorous certificate which validates whether a given MPC feedback law avoids a given set of unsafe states. It will be also shown that such verification procedure can be employed as an another memory reduction technique.

Verification of A-Posteriori Quantized MPC

Control under finite precision arithmetic is a very important research field since nowadays majority of the control policies are implemented on digital platforms (which implies usage of A/D and D/A conversions). Moreover, it should be mentioned that there even exist systems with naturally quantized control behavior, e.g. automatic transmissions, whose control precision is strictly limited by a finite number of gears that are given by the systems construction. Devising an MPC strategy for such systems is not an easy task. The solution to this problem usually leads to complex optimization problems which require increased computation power and an appropriate solver. Therefore, when one aims to apply these feedback laws into lower-level industrial hardware, one can easily come across with some implementation issues, which will make the implementation impossible.

In this thesis we tackle this problem by proposing a verification technique. Particularly, it is shown that MPC policy can be constructed without any knowledge of the rounding-based quantizer (that is present in the closed loop) and then one can use multiparametric programming as a tool for an a-posteriori verification whether such MPC policy provides desired control properties (namely recursive feasibility, closed-loop asymptotic stability or performance guarantees) even in the presence of quantization. The main motivation behind this direction is that if we are able to certify all necessary control properties of MPC policies, then one can safely employ these less complex MPC feedback laws, which have reduced implementation requirements.

This verification technique assumes that we are given an explicit representation of the MPC law $\mu(\cdot)$ as a PWA function of the state, along with the information of the rounding-based quantizer, i.e. the number and values of all quantization

levels q_i. Based on this knowledge the method proceeds as follows. Firstly, we devise an a-posteriori quantized control law $\mu_{\mathrm{q}}(\cdot)$, which characterizes how the real-valued control inputs of $\mu(\cdot)$ is being rounded to the nearest quantization level q_i, via employing the technique known as Voronoi diagrams. It is shown that $\mu_{\mathrm{q}}(\cdot)$ takes a form of a piecewise constant (PWC) function and is defined over a polytopic partition that consists of $D \geq d$ regions, where d denotes the number of quantization levels. Next, we suggest to construct performance bounds $\overline{V}$ and $\underline{V}$ as to describe desired control properties, such as asymptotic closed-loop stability, recursive feasibility and bounded deterioration of performance. And finally, in the last step, we synthesize a rigorous certificate that verifies whether $\overline{V} \leq \mu_{\mathrm{q}}(\cdot) \leq \underline{V}$ is satisfied, i.e., whether the quantized feedback law meets all performance criteria. If a positive certificate is obtained, then we have a guarantee that the real-valued controller $\mu(\cdot)$ provides all the examined control properties and can be hence implemented in the closed loop, where the investigated rounding-based quantizer is located. On the other hand, if a negative certificate is obtained, then certain property is known to be violated, e.g. the MPC policy $\mu(\cdot)$ might yield such control inputs, which after the quantization effect violate the control constraints. In such a case, implementation of $\mu(\cdot)$ would represent a risk and thus it is then convenient to re-design and/or re-tune the MPC policy.

Safety Verification in MPC

In the recent years, the safety aspect takes an increasing attention in control design. This trend has multiple causes. The most important reasons are to simply avoid any hazardous situations, the outcome of which might lead to a great mechanical stress, to impairment of product or even to explosions. Therefore, in order to minimize economical losses and preserve human life, one needs to impose these criteria into the control design. However, this task is not easy at all. The difficulty stems from the fact that including certain safety properties directly into the optimization problem often leads to non-convex formulations that are computationally expensive to solve in real time. Therefore, in order to meet limitations of the industrial hardware one needs to take additional measures and mitigate the computational requirements of MPC to an acceptable level. In this thesis we show one direction how to do that.

As in the previous approach, we suggest to omit safety constraints (e.g. obstacle

avoidance constraints) from the MPC optimization problem, hence to keep it convex. Subsequently, we propose a non-conservative certification procedure which a-posteriori examines enforcement of all these neglected safety constraints. Particularly, we verify whether the MPC controller is designed such that it forces the states of the controlled system, initialized from a set of known initial conditions, to avoid all unsafe states. In another words, whether the unsafe situation cannot be reached with the simple MPC feedback law in the closed loop.

We show that this certification can be done without the need of having the analytical solution of MPC in hand. Instead, we exploit the Karush-Kuhn-Tucker (KKT) conditions which implicitly encode the optimal control inputs. Under mild assumptions this verification task can be carried out for infinite number of time steps, i.e. ad-infimum. The final optimization problem takes a form of a mixed integer linear program (MILP), the solution of which then indicates if the simple MPC feedback law is well designed and provides guarantees of safety, or if it is poorly designed and violates the safety criteria.

pH Control in a Chemical Vessel

Maintaining a specific value of pH represents a very important task for many processes. The most common applications can be found in food processing, wastewater treatment, pulp and paper production, and last but not least in chemical industries. It has to be stressed that majority of process plants in various industries produce secondary unwanted products, such as wastewater, which must be neutralized before they are reused or discharged into rivers. Biochemical experiments are another application where maintaining of a specific pH value is needed. Here, unfavorable pH conditions negatively affect entire experiments. The value of pH plays also a vital role in medical research and medicine preparation since a vast majority of all drug preparation requires specific pH conditions. Needless to say, with ever increasing pressure on product quality, process efficiency and environmental protection, good pH control is highly desired.

In this thesis, we aim to provide a real application of MPC-based pH control. The targeted process consists of two pumps which feed acetic acid and sodium hydroxide solutions into the mixing vessel, where the chemical reaction between these two reactants takes place. By keeping inflow of the acid solution constant, our objective is to manipulate the flow rate of the base solution such that effluent

from the system will attain specific value of the pH. The main challenge here is to design a control strategy that will be able to handle the highly non-linear behavior of this process and will exhibit a good control performance.

We approach this problem by designing a model of the system. Here, a black-box identification approach is chosen, i.e. the model is obtained based on the input and output responses of the controlled process. Subsequently, a reference governor control strategy is proposed, where MPC shapes references for PID controller in an optimal fashion, i.e. such that technological constraints are satisfied and specified performance criterion is optimized. Such a control scheme is often used in chemical and petrochemical industries with the goal of increasing the overall production quality and quantity.

MPC Application to Wind Turbine Generator

Potential of wind as a power source was recognized thousand years ago. The first instance of wind powering a machine in history was a windmill operating organ that was built by Heron of Alexandria in the 1st century AD. Since then, people have exploited wind power e.g. in windmills in order to automate grain grinding and water pumping, or in sailboats to mitigate labor demands. Recently, this eye-appealling and free power source has experienced an increased popularity by providing the answer for still increasing world energy consumption, as well as to climatic changes, via the use of wind turbines. Wind turbines are devices used to transform kinetic energy of the wind into mechanical rotation energy, which is then converted into electric energy in the generator. During the last two decades a lot of attention was dedicated to development of wind turbines, where the main goals were to maximize energy production, improve energy quality and minimize costs of installation and maintenance (e.g. by prolonging the life cycle of wind turbine).

In this thesis, we will address the problem of implementing fast and safe control strategy into the wind turbine. Particularly, we aim to construct MPC strategy for rotor-flux-based field-oriented control (FOC) with voltage-controlled converter, where control voltages are subjected to safety criteria. We show that to achieve this goal is not an easy task, especially when circular safety constrains have to be satisfied and rapid sampling period of 0.4 milliseconds has to be preserved. The final controller will be based on explicit MPC policy with applied move-blocking and disturbance modeling techniques.

Goals of the Thesis

This work contributes to the field of MPC and aims to find new possibilities of its fast and less memory demanding implementation. The main objectives can be summarized into three principal directions as follows:

- Novel memory reduction techniques which mitigate memory consumption of explicit model predictive controllers.
- Performance analysis of model predictive feedback laws, with the objective to provide:
  - Rigorous conditions under which a given explicit MPC controller provides guarantees of closed-loop stability, recursive constraint satisfaction and bounded performance deterioration when its control commands are quantized.
  - Non-conservative reachability procedure, which determines whether an implicitly defined MPC feedback law forces the closed-loop system to avoid unsafe states from a particular set of initial conditions.
- Implementation of model predictive control on real processes. Systems of interest are:
  - pH reactor, where chemical reaction between acetic acid and sodium hydroxide takes place,
  - application of MPC to a wind turbine generator.

Complexity and memory reduction techniques were published in:

1. Holaza, J. – Takács, B. – Kvasnica, M. : Synthesis of Simple Explicit MPC Optimizers by Function Approximation. In *Proceedings of the 19th International Conference on Process Control*, štrbské Pleso, Slovakia, pp. 377–382, 2013.
2. Takács, B. – Holaza, J. – Kvasnica, M. – Di Cairano, S. : Nearly-Optimal Simple Explicit MPC Regulators with Recursive Feasibility Guarantees. In *IEEE Conference on Decision and Control*, Florence, Italy, pp. 7089–7094, 2013.

3. Holaza, J. – Takács, B. – Kvasnica, M. : Simple Explicit MPC Controllers Based on Approximation of the Feedback Law. In *ACROSS Workshop on Cooperative Systems*, Zagreb, Croatia, pp. 48–49, 2014.

4. Holaza, J. – Takács, B. – Kvasnica, M. – Di Cairano, S.: Nearly optimal simple explicit MPC controllers with stability and feasibility guarantees. In *Optimal Control Applications and Methods*, vol.6, 2015.

Techniques devoted to performance analysis were published in:

1. Holaza, J. – Takács, B. – Kvasnica, M. : Verification of Performance Bounds for A-Posteriori Quantized Explicit MPC Feedback Laws. In *Preprints of the 19th IFAC World Congress Cape Town*, Cape Town, South Africa, pp. 1035–1040, 2014.

2. Holaza, J. – Takács, B. – Kvasnica, M. : Safety Verification of Implicitly Defined MPC Feedback Laws. In *European Control Conference*, Linz, Austria, pp. 2552–2557, 2015.

Results from the practical applications are in process of publishing.

Part I

Theoretical Basis

Chapter 2

Mathematical Optimization

A mathematical optimization problem, or simply an optimization problem, is the search of the best possible solution from a set of all feasible candidates. This technique is widely used in numerous fields e.g. in economics, or in mechanical and control engineering. It provides the answer e.g. how to decrease a waste in the production of some products (thus to increase income of manufacture) or how to find the most fuel-efficient path from one place to another.

An optimization problem is composed of two parts and in general can be formulated as

$$J^\star = \inf_x f_0(x) \tag{2.1a}$$

$$\text{s.t. } x \in \mathcal{S} \subseteq \mathcal{X}. \tag{2.1b}$$

The first part (2.1a) is called the objective function (or a cost function) with $f_0 : \mathbb{R}^n \mapsto \mathbb{R}$, which assigns to each choice of the optimization variable $x = (x_1, x_2, \ldots, x_n)^T$ its cost $f_0(x)$. The second one (2.1b) are the constraints that specify the space of all admissible optimization variables $\mathcal{S} \subseteq \mathbb{R}^n$, where $\mathcal{X} \subseteq \mathbb{R}^n$ denotes the domain of the objective function.

The solution $x^\star$ to (2.1), or simply the optimizer, is given as the vector $x^\star = (x_1^\star, x_2^\star, \ldots, x_n^\star)^T$, where the objective function $J^\star = f_0(x^\star)$ acquires its infimum with respect to constraints (2.1b). In general we can classify the solution to (2.1) into three categories:

Case 1: Problem is bounded, when $J^\star > -\infty$ and $x^\star \in \mathcal{S}$.

Case 2: Problem is unbounded (from below), when $J^\star = -\infty$. This situation occurs when set $\mathcal{S}$ does not stop vector $x^\star$ to lead $J^\star$ into infinity, or when the problem is unconstrained, i.e., $\mathcal{S} = \mathbb{R}^n$.

Case 3: Problem is infeasible, when $J^\star = +\infty$. This is due to the fact that the set $\mathcal{S}$ is empty.

Furthermore, optimization problem is called a feasibility problem if is given by

$$J^\star = \inf_x c \tag{2.2a}$$

$$\text{s.t. } x \in \mathcal{S} \subseteq \mathcal{X}, \tag{2.2b}$$

where c is a constant $c \in \mathbb{R}$. The purpose of (2.2) is only to find out whether there exist any vector x that satisfies constraints $\mathcal{S}$. Hence the result of this problem can by either $J^\star = c$ if (2.2) is feasible, or $J^\star = \infty$ if (2.2) is infeasible.

2.1 Hierarchy of Optimization Problems

Optimization problems can be divided into multiple classes. We can classify them on the bases of several indicators, e.g. type of the objective function, constraints and optimization variables. A comprehensive overview can be seen in Figure 2.1. Nevertheless, all optimization problems can be sorted into two main categories. They are either convex or non-convex optimization problems. Convex optimization problems are characterized by a convex objective function f_0 and a convex constraints set $\mathcal{S}$. These optimization problems are widely adopted due to their natural advantages:

1. Solution to a convex problem either always yields a global optimum $f_0(x) \geq f_0(x^\star)$ if the problem is feasible.
2. Convex problems can be solved relatively easily compared to non-convex ones.
3. There is a wide variety of mature solvers for this type of problems.

On the other hand, non-convex optimization problems have either a non-convex domain and/or a nonconvex objective function. Nonconvex problems may feature multiple local optima, which renders the finding of the global optimum a time consuming task.

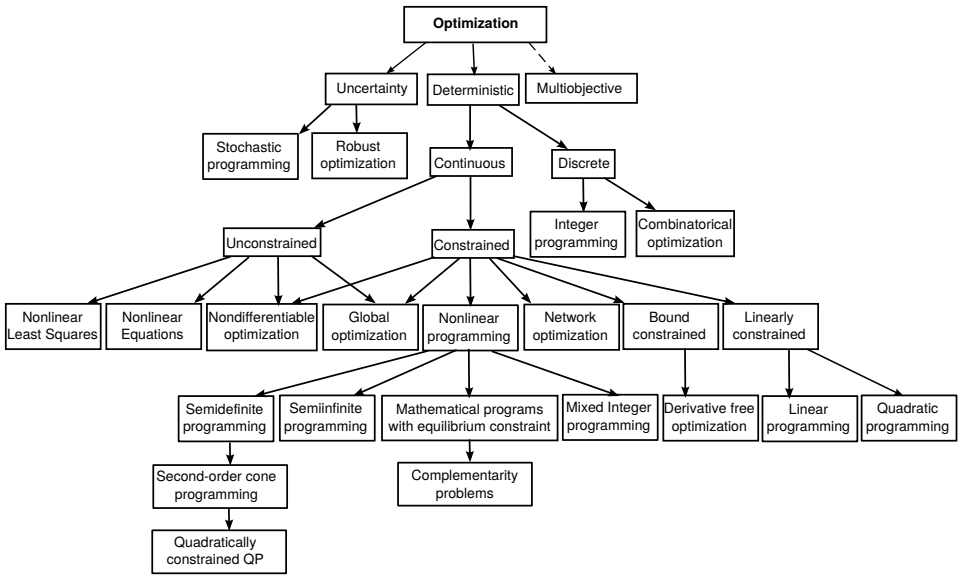
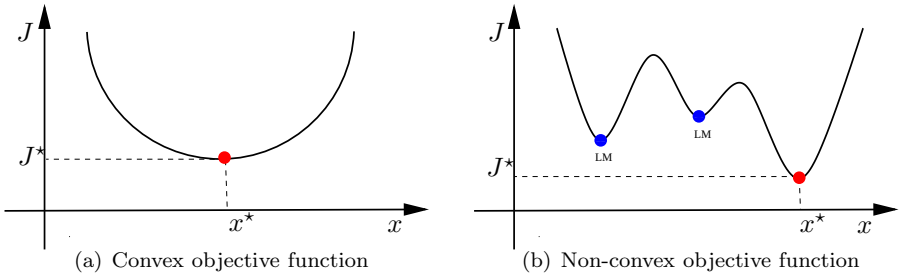


Figure 2.1: Hierarchy of optimization problems (NEOS, 2014).

The basic difference between convex and non-convex optimization can be seen in Figure 2.2. Here, it is evident that in the case of convex optimization, there is only one global optimum. But in the non-convex case one can find even multiple optima, while only one being global. Evidently, in this case it is very important to provide to the solver such initial condition, which is relatively close to the global optimum. Otherwise it may take a longer time to find the global optimum (if even the solver will find it at all).



(a) Convex objective function (b) Non-convex objective function

Figure 2.2: Comprehension of two different types of optimization problems. Here blue circles denote local minima and red circles global minima.

2.2 Convex Optimization Problems

This section will be concerned with fundamentals regarding convex functions and convex sets (constraints), since such requirements have to be met in convex optimization. An excellent overview on this topic can be found in Boyd and Vandenberghe (2004).

2.2.1 Objective Function

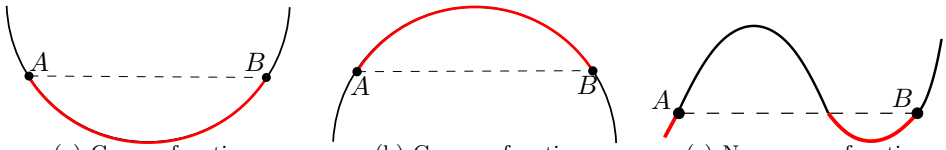
Objective functions play a very important role in optimization problems, as they specify how each optimization variable is going to be penalized. There are many possibilities how one can penalize optimization variables. One can use linear cost (e.g. $c^T x$), quadratic cost (e.g. $x^T C x$), logarithmic cost (e.g. $log(x)$), and so on. We need, however, keep in mind that the character of these costs determine the convexity of the objective function and thus the type of the optimization problem (see Figure 2.2). As we have pointed out, here we will aim at such functions that lead to convex optimization problems.

Definition 2.2.1 (Convex function) *A function $f : \mathcal{S} \mapsto \mathbb{R}$ is called convex if $\mathcal{S}$ is a convex set and for any two optimization variables $x_1, x_2 \in \mathcal{S}$ following relation holds:*

$$f(\theta x_1 + (1-\theta)x_2) \leq \theta f(x_1) + (1-\theta) f(x_2), \tag{2.3}$$

for all $\theta \in [0, 1]$.

Convexity can be also examined geometrically, where an example is depicted in Figure 2.3. Here two points x_1 and x_2 are chosen such that $x_1 \neq x_2$. They define the line segment from $[x_1, f(x_1)]$ to $[x_2, f(x_2)]$. Then the function $f(x)$ is convex if its graph lies under this chord (Figure 2.3(a)). On the other hand, if it does not lie under this line segment, then the function $f(x)$ is non-convex (Figure 2.3(c)). Naturally, if sign $\leq$ is replaced by $\geq$ in (2.3), then the fulfillment of the modified expression indicates concavity (Figure 2.3(b)). Moreover, in a case of affine function, the equation (2.3) always holds for both signs ($\leq$ and $\geq$). In other words, affine (as well as linear) functions are both convex and concave.



(a) Convex function (b) Concave function (c) Non-convex function

Figure 2.3: Graphs of convex, concave and non-convex functions. Here, for brevity, points $A = [x_1, f(x_1)]$ and $B = [x_2, f(x_2)]$ are marked as black circles and the line segment $(\bar{AB})$ is donated by black dashed line. The red segment of each function indicates the inequality (2.3).

2.2.2 Constraints

Constraints have a significant impact in an optimization problem, hence in control of real technological processes or technical systems, where they restrict the domain $\mathcal{X}$ of (2.1) to an admissible set of optimization variables $\mathcal{S}$. Application of constraints in a control design has a lot of important aspects. They are used for a better representation of physical systems (e.g. input saturation), to ensure stable control (e.g. constraints in the form of end-stabilizing constraints), and last but not least for tuning controller parameters to achieve better quality of control. In terms of character constraints can be separated into two categories:

Hard constraints

This type of conditions must be always met, otherwise the optimization problem (2.1) would be infeasible (and the control input would not be provided). A representative sample of hard constraints are physical restrictions, an example of which can be represent by a valve (that can be opened only between the interval $[0\%, 100\%]$) or by a gear in a car (where the gear lever can not exceed level that is higher than the design allows).

Soft constraints

This type of constraints may be violated, but the level of their violation is reflected in the objective function, hence minimized. An example of soft constraints is safety restrictions (such as speed limitations, operating temperature, operating noise, and many others), or all types of comfort restrictions.

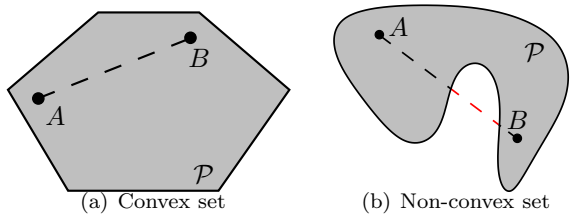
Recalling that in convex optimization both objective function and constraint must be convex. Since the convexity of a function we have already examined in Sec-

tion 2.2.1, now we will take a closer look at convex sets (Fukuda., 2004; Gallini, 2014; Grunbaum, 2000; Webster, 1995; Ziegler, 1995). Also we would like to note that most of the presented techniques are already embedded e.g. in the Multi-parametric toolbox (MPT) (Herceg et al., 2013a).

Definition 2.2.2 (Convex set) *A set $\mathcal{S}$ is convex if for any $x_1, x_2 \in \mathcal{S}$, $\lambda, \mu \geq 0$, $\lambda + \mu = 1$*

$$\lambda x_1 + \mu x_2 \in \mathcal{S}. \tag{2.4}$$

Graphically can be definition 2.2.2 interpreted as follows. Let us choose two (different) points $x_1, x_2 \in \mathcal{S}$, such that $x_1 \neq x_2$. Then a set $\mathcal{S}$ is convex if and only if any convex combination of this points lies in $\mathcal{S}$. Simply put, the line segment created by this two points x_1 and x_2 must be contained in the set $\mathcal{S}$. Figure 2.4 illustrates an example of a convex and non-convex set.



(a) Convex set (b) Non-convex set

Figure 2.4: Simple exmaples of convex and non-convex sets. Here, for brevity, points $A = x_1$ and $B = x_2$ are marked as black circles and the line segment ($\overline{AB}$) is denoted by dashed line. The red segment denotes violation of convexity.

In practical applications, we can often encounter with convex constraints, which are easily solvable, and can be defined as e.g. polyhedra, polytopes, simplices, ellipsoids and other sets.

Definition 2.2.3 (ϵ-ball) *We denote the ϵ-ball $\in \mathbb{R}^n$ with radius ϵ and centered around $x_c \in \mathbb{R}^n$ by $\mathcal{B}_\epsilon(x_c) = \{X \in \mathbb{R}^n \mid \|x - x_c\| < \epsilon\}$, where $\|\cdot\|$ denotes any norm.*

Definition 2.2.4 (Closed set) *We say that a set $\mathcal{P} \subseteq \mathbb{R}^n$ is closed if*

$$\forall x \notin \mathcal{P} \subseteq \mathbb{R}^n, \exists \epsilon > 0 : \mathcal{B}_\epsilon(x_c) \cap \mathcal{P} = \emptyset. \tag{2.5}$$

Definition 2.2.5 (Bounded set) *We say that a set $\mathcal{P} \in \mathbb{R}^n$ is bounded if it is contained in some $\epsilon-$ball of finite radius, i.e.*

$$\exists \epsilon < \infty : \mathcal{P} \subseteq \mathcal{B}_\epsilon(x_c). \tag{2.6}$$

Definition 2.2.6 (Polyhedron) *Polyhedron $\mathcal{P}$ in $\mathbb{R}^n$ denotes a convex and closed set defined as an intersection of a finite number of $n_\mathcal{H}$ closed affine half-spaces $a_i^T x \leq b_i$, $a_i \in \mathbb{R}^n$, $b \in \mathbb{R}$, $\forall i = 1, \ldots, n_\mathcal{H}$. A compact notation of a polyhedron is*

$$\mathcal{P} = \{x \in \mathbb{R}^n \mid Ax \leq b\}, \tag{2.7}$$

where $A \in \mathbb{R}^{n_\mathcal{H} \times n}$, $b \in \mathbb{R}^{n_\mathcal{H}}$.

Definition 2.2.7 (Polytope) *Set $\mathcal{P}$ is called a polytope if it is a bounded polyhedron.*

Definition 2.2.8 (Vertex representation of a polytope) *A polytope $\mathcal{P} \subset \mathbb{R}^n$ can be defined in its vertex representation as*

$$\mathcal{P} = \{x \mid x = \textstyle\sum_i \lambda_i v_i,\ 0 \leq \lambda_i \leq 1,\ \sum_i \lambda_i = 1\}, \tag{2.8}$$

where $v_i \in \mathbb{R}^n$, $\forall i = 1, \ldots, M$ are vertices of the polytope.

Remark 2.2.9 *Note that Definition 2.2.8 can be viewed at as a convex hull of all vertices, i.e. $\mathcal{P} = conv(V)$, where $V = \{v_1, \ldots, v_M\}$ (see definition 2.2.14)*

For the brevity, Definition 2.2.7 will refer to $\mathcal{H}$-polytope (Figure 2.5(b)), and Def. 2.2.8 to $\mathcal{V}$-polytope (Figure 2.5(c)). Another important category of convex sets are simplices (Figure 2.5(d)), which play an important role e.g. in triangulation techniques.

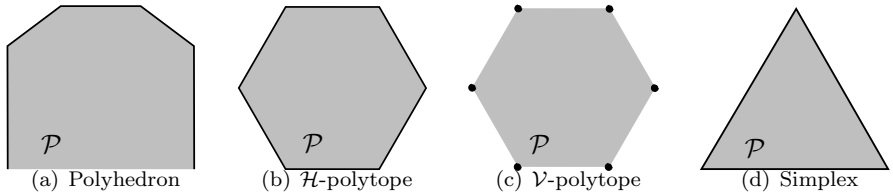
Definition 2.2.10 (Simplex) *A n-dimensional simplex, or simply n-simplex, is a polytope $\mathcal{P}$ in $\mathbb{R}^n$, which is a convex hull (see definition 2.2.14) of its $n+1$ vertices $\{v_1, \ldots, v_{n+1}\}$*

$$\mathcal{P} = conv(v_1, \ldots, v_{n+1}). \tag{2.9}$$

Definition 2.2.11 (Set collection) *We say that a set $\mathcal{S} \subseteq \mathbb{R}^n$ is a set collection if it collects a finite number of sets $\mathcal{S}_i \subseteq \mathbb{R}^n$, i.e.*

$$\mathcal{S} = \{\mathcal{S}_i\}_{i=1}^{Ns}, \tag{2.10}$$

where $i = 1, \ldots, Ns$ and $Ns < \infty$.



(a) Polyhedron (b) $\mathcal{H}$-polytope (c) $\mathcal{V}$-polytope (d) Simplex

Figure 2.5: Figure depicts four convex constraints (sets).Figure 2.5(a) shows an opened polyhedron. In Figure 2.5(b) and Figure 2.5(c) one can see polytopes in their half-space ($\mathcal{H}$) and vertex ($\mathcal{V}$) representations, respectively. Further, 2.5(d) illustrates a 2-dimensional simplex, hence 2-simplex.

Definition 2.2.12 (Polytopic partition) *Polytopic partition of a polytope Ω is given by the collection of M polytopes $\{\mathcal{R}_i\}_{i=1}^{M}$ if:*

1. $\Omega = \cup_i \mathcal{R}_i$,
2. $\text{int}(\mathcal{R}_i) \cap \text{int}(\mathcal{R}_j) = \emptyset$, $\forall i \neq j$. *def:partition*

We call each polytope of the collection a region *of the partition.*

Properties of Convex Sets

We say that $i-$th half-space of a polyhedron in (2.7), i.e. $a_i^T x \leq b_i$, is *redundant* if $\mathcal{P} = \{x \in \mathbb{R}^n \mid A_{\setminus i} x \leq b_{\setminus i}\}$, where $A_{\setminus i}$, $b_{\setminus i}$ denote matrices A, b, without its $i-$th row. Or, in another words, $\mathcal{P}$ remains the same even if the redundant half-space is removed from (2.7). Moreover, $\mathcal{P}$ has *minimal* representation if it has no redundant half-spaces.

We say that $x \in \mathcal{P}$ is an interior point of $\mathcal{P}$ if there exists $\mathcal{B}_\epsilon(x_c) \in \mathcal{P}$ with $\epsilon > 0$. Subsequently, an interior of $\mathcal{P}$, i.e. $\text{int}(\mathcal{P})$, is the collection of all interior points.

We refer $\mathcal{P}$ to be full-dimensional if $\exists \mathcal{B}_\epsilon(x_c) \subset \mathcal{P} \subset \mathbb{R}^n$ with $\epsilon > 0$.

Next let polytope $\mathcal{P} \subset \mathbb{R}^n$ be in its minimal representation, full-dimensional and denoted by $n_\mathcal{H}$ half-spaces e.g. (2.7). Then we have that $\mathcal{P}$ is bounded by another full-dimensional polytopes in $\mathbb{R}^{n-k}$ that are generally referred as faces $\mathcal{F}$ denoted as

$$\mathcal{F} = \mathcal{P} \cap \{x \in \mathbb{R}^n \mid Ax = b\}.$$

Moreover, faces $\mathcal{F}$ based on their dimension are called $0-$vertices, $1-$edges, $(n-2)$-ridges and $(n-1)$-facets.

We have that $i-$th vertex of a polyhedron in (2.8), i.e. v_i, is redundant if $\mathcal{P} = \text{conv}(V_{\setminus i})$. Minimal representation of (2.8) is when there are no redundant vertices.

Operations With Sets

Generally we can still arise with constraints that are defined as non-convex sets. As an illustrative example can serve a squared parking place where one car is already parked. Subsequently, it does not matter where this car parks, next car that would come to park will enter a non-convex parking place. Therefore this second car should then operate with non-convex set (constraint) that lead to a non-convex optimization problem.

Such a non-convex set can be obtained by removing one set (car) from another (parking place). This technique is referred as a set difference.

Definition 2.2.13 (Set difference) *Let $\mathcal{P}$ and $\mathcal{Q}$ be two polytopes, then the set difference between these two polytopes can be defined as*

$$\mathcal{P} \setminus \mathcal{Q} = \{x \mid x \in \mathcal{P}, x \notin \mathcal{Q}\}. \tag{2.11}$$

Note that generally a set difference $\mathcal{R} = \mathcal{P} \setminus \mathcal{Q}$ yields a non-convex set $\mathcal{R}$, yet it can be defined as a collection of finite polytopes, i.e. $\mathcal{R} = \cup_{i=1}^{k}$. Moreover, since $\mathcal{P}$ and $\mathcal{Q}$ are closed sets, their set difference $\mathcal{R}$ is not closed set. But, for simplicity, we consider its closure.

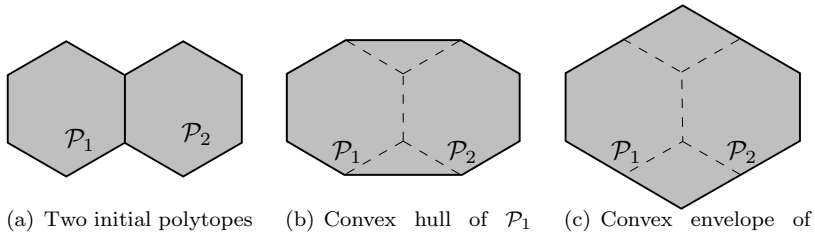
Since non-convex sets are extremely hard to handle in optimization problems, one can still use a technique (e.g. convex hull and envelope) that will convert such a set into its convex (approximated) form.

Definition 2.2.14 (Convex hull) *Let $V = (v_1, \ldots, v_M)^T \subset \mathbb{R}^n$ be a finite vector of points $v_i \in \mathbb{R}^n$. Then the convex hull is the smallest convex set containing all points V and is given by*

$$conv(V) = \{\textstyle\sum_{i=1}^{v_M} \lambda_i v_i \mid \lambda_i \geq 0, \sum_{i=1}^{v_M} \lambda_i = 1\}. \tag{2.12}$$

Consider next union of polytopes $\mathcal{R} = \cup_{i=1}^{M} \mathcal{P}_i$. The convex hull of $\mathcal{R}$ is a polytope defined as

$$conv(\mathcal{R}) = \{\textstyle\sum_{i=1}^{M} \lambda_i x_i \mid x_i \in \mathcal{P}_i, \lambda_i \geq 0, \sum_{i=1}^{M} \lambda_i = 1\}. \tag{2.13}$$



(a) Two initial polytopes $\mathcal{P}_1$ and $\mathcal{P}_2$

(b) Convex hull of $\mathcal{P}_1$ and $\mathcal{P}_2$

(c) Convex envelope of $\mathcal{P}_1$ and $\mathcal{P}_2$

Figure 2.6: Figure illustrates convex hull 2.6(b) and convex envelope 2.6(c) of two polytopes 2.6(a).

Note that the opposite operation to convex hull is technique referred as *vertex enumeration*, e.g. $V = \text{vert}(\mathcal{P})$, see (Borrelli et al., 2016; Fukuda., 2004). This technique is used e.g. to transform polytope from $\mathcal{H}$-representation into its $\mathcal{V}$-representation.

Definition 2.2.15 (Convex envelope) *A convex envelope of two $\mathcal{H}$-polyhedra $\mathcal{P} = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ and $\mathcal{Q} = \{x \in \mathbb{R}^n \mid Cx \leq d\}$, as in 2.2.6, is an $\mathcal{H}$-polyhedra defined as*

$$env(\mathcal{P}, \mathcal{Q}) = \{x \in \mathbb{R}^n \mid Ax \leq b,\ Cx \leq d\}. \tag{2.14}$$

The convex hull as well as the convex envelope technique are compared in Figure 2.6, where union of two polytopes is considered. Here, one could notice the difference between these two methods, in terms of volumes of the final polytope created via the convex hull and the envelope technique, respectively.

Next, we define two additional techniques. First one is called projection, which essentially transforms objects (constrains) to their lower dimension representation, and the second one is triangulation, which is used to tessellate objects into multiple simplices.

Definition 2.2.16 (Projection) *Consider vectors $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ and a polytope $\mathcal{P} = \{[x^T y^T]^T \mid A^x x + A^y y \leq b\}$. Then the projection of $\mathcal{P} \subset \mathbb{R}^{n+m}$ into $x \in \mathbb{R}^n$ can be defined as*

$$proj_x(\mathcal{P}) = \{x \in \mathbb{R}^n \mid \exists y \in \mathbb{R}^m, A^x x + A^y y \leq b\}. \tag{2.15}$$

Definition 2.2.17 (Triangulation) *Triangulation of a polytope $\mathcal{P} \subset \mathbb{R}^n$ is a set of k n-simplices*

$$\triangle_k = triangulate(\mathcal{P}), \tag{2.16}$$

such that $\cup_k \triangle_k = \mathcal{P}$ and $int(\triangle_i) \cap int(\triangle_j) = \emptyset$, $\forall i \neq j$.

2.2.3 Linear Programming

There is a wide classification of optimization problems which are categorized based on their formulation, particularly structure of the objective function (2.1a) and constraints 2.1b. An important class of optimization problems is linear programming (LP), where the cost function and the constraints of (2.1) are linear. Therefore, from Section 2.2.1 and Section 2.2.2, we have that LP is a convex optimization problem that can be in general stated as

$$J^\star = \min_x c^T x \tag{2.17a}$$

$$\text{s.t. } Ax \leq b, \tag{2.17b}$$

where $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. There are two common forms of LPs. The first one includes both equality and inequality constraints:

$$J^\star = \min_x c^T x \tag{2.18a}$$

$$\text{s.t. } Ax \leq b \tag{2.18b}$$

$$A_{eq} x = b_{eq}, \tag{2.18c}$$

where $A_{eq} \in \mathbb{R}^{p \times n}$ and $b_{eq} \in \mathbb{R}^p$. In the second formulation, only equality constraints are present, along with trivial inequalities:

$$J^\star = \min_x c^T x \tag{2.19a}$$

$$\text{s.t. } A_{eq} x = b_{eq} \tag{2.19b}$$

$$x \geq 0. \tag{2.19c}$$

The reason, why one should convert the form of LP is that there exist a wide variety of solvers which can solve LP efficiently in a certain form. This applies even if the number of variables is being increased (e.g. as it is in transformation from a form of (2.18) to (2.19), where the number of variables raised from n to $2n + p$). It should be noted that there already exist multiple programs that automatically

adjust formulation of an optimization problem into its appropriate form based on the used solver (see e.g. (Löfberg, 2004)).

Solution Properties of LP

Consider optimization problem given as (2.17), where the intersection of (2.17b) denotes polyhedron $\mathcal{P}$. Furthermore, denote by $J^\star$ the optimal value of the objective function and by $X^\star$ the set of optimal solutions to (2.17). Then we can encounter four different scenarios:

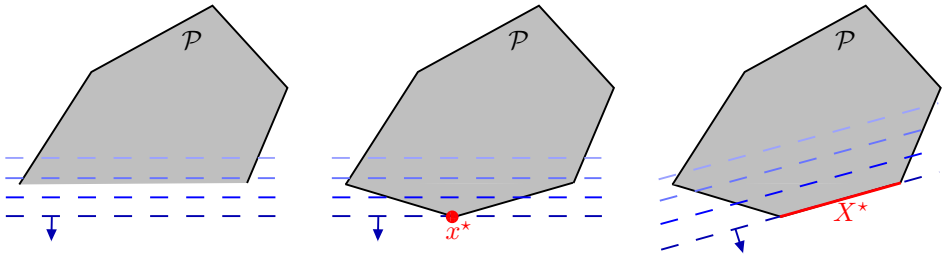
Case 1: The problem is unbounded (from below), then $J^\star = -\infty$.

Case 2: The problem is bounded, then $J^\star > -\infty$, with unique solution. Hence $X^\star$ is a singleton.

Case 3: The problem is bounded ($J^\star > -\infty$), but with multiple optima $X^\star$. Here $X^\star$ is a set of uncountable $x^\star$ in $\mathbb{R}^n$. This is due to a fact that objective function is parallel to the constraint, where optimum has been found.

Case 4: The problem is infeasible, $J^\star = \infty$, which is caused by the polyhedron $\mathcal{P}$ being an empty set.

First three scenarios are illustrated in Figure 2.7 by using two-dimensional examples.

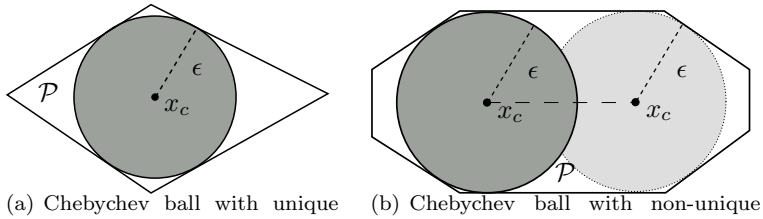


(a) Unbounded LP problem

(b) Bounded LP problem with unique optimizer.

(c) Bounded LP problem with non-unique optimizer.

Figure 2.7: Figure depicts three different results of LP problems. Here the blue dashed lines represent contours of the objective function, the blue arrow the direction of minimization and $\mathcal{P}$ considered polyhedron.



(a) Chebychev ball with unique center.

(b) Chebychev ball with non-unique center.

Figure 2.8: Figure shows solution of the Chebychev LP problem in (2.20) for two distinguish polyhedra $\mathcal{P} \subset \mathbb{R}^2$. Here the radius of the ball is $\epsilon > 0$, while the center may attain various values.

Chebychev ball

As one of the suitable examples of LP can be chosen the Chebychev approximation problem, which is also referred to as the Chebychev ball. The objective is to inscribe a ball $\mathcal{B}_\epsilon(x_c) \subset \mathcal{P}$ with the largest radius ϵ and with the center x_c into a polyhedron $\mathcal{P} \subset \mathbb{R}^n$ as in (2.2.6), which corresponds to the intersection of n_{a_i} half-spaces $a_i^T x \leq b_i$, $i = 1, \ldots, n_{a_i}$. The aforementioned task can be then formulated as an LP in the form of:

$$\max_{x_c, \epsilon} \ \epsilon \tag{2.20a}$$

$$\text{s.t. } a_i x_c + \epsilon \|a_i\|_2 \leq b_i, \ i = 1, \ldots, n_{a_i} \tag{2.20b}$$

with vectors $a_i \in \mathbb{R}^n$ and $b_i \in \mathbb{R}$, defining the set $\mathcal{P}$. By solving (2.20) one can obtain three results:

- If the optimal radius of the ball $\epsilon^\star > 0$, then the polyhedron $\mathcal{P}$ is full-dimensional (see Figure 2.8).
- If $\epsilon^\star = 0$, then the polyhedron $\mathcal{P}$ is lower-dimensional, but not empty (e.g. in case of $n = 2$ polyhedron can be a line segment or a point).
- If $\epsilon^\star < 0$, then the polyhedron $\mathcal{P}$ is an empty set.

Remark 2.2.18 *The Chebychev ball problem* (2.20) *can have actually multiple centers x_c, for the same objective function value ϵ, radius respectively. This is due to the fact that to the currently investigated polyhedron $\mathcal{P}$ one can write $\mathcal{B}(x_c, \epsilon)$*

several times. For example as shows Figure 2.8(b), an infinite number of circles can be written into $\mathcal{P}$ (along the dashed line).

2.2.4 Quadratic Programming

An optimization problem is called a quadratic program (QP), if (2.1) has linear constraints and a quadratic objective function. In general, convex QPs can be formulated as

$$J^\star = \min_x \ \tfrac{1}{2}x^T H x + q^T x + c \tag{2.21a}$$

$$\text{s.t. } Ax \leq b, \tag{2.21b}$$

or with constraints in a form of equalities as

$$J^\star = \min_x \ \tfrac{1}{2}x^T H x + q^T x + c \tag{2.22a}$$

$$\text{s.t. } Ax \leq b \tag{2.22b}$$

$$A_{eq}x = b_{eq}, \tag{2.22c}$$

where $x \in \mathbb{R}^n$, $H = H^T \succ 0 \in \mathbb{R}^{n\times n}$, $q \in \mathbb{R}^n$, $c \in \mathbb{R}$, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $A_{eq} \in \mathbb{R}^{p\times n}$ and $b_{eq} \in \mathbb{R}^p$. Next, we will show how to reformulate QP from (2.22) into (2.21), i.e. how to eliminate equality constrains. This technique is commonly used in the formulations of advanced control strategies, which we will discuss later. The reversed direction will be omitted as it is straightforward procedure. Particularly, we can define $A_{eq} = \mathbf{0}$ and $b_{eq} = \mathbf{0}$, where $\mathbf{0}$ is a zero matrix of appropriate dimensions.

Eliminating Equality Constraints

By comparing QP forms (2.22) and (2.21), one can see the main challenge is to eliminate equality constraints. The most straightforward way would by to rewrite this equality constraint into two inequality constraints, i.e. substitute $A_{eq}x = b_{eq}$ by $A_{eq}x \leq b_{eq}$ and $-A_{eq}x \leq b_{eq}$. The shortcoming, however, is that this technique may lead to a bad numerical conditioning, thus can jeopardize accuracy in solvers. To workaround this problem one can parameterize the solution that will lead into a new optimization vector $\lambda \in \mathbb{R}^{n-p}$ with decreased number of elements, compared to the original vector $x \in \mathbb{R}^n$, by exactly the number of equality constraints p.

To proceed we will firstly find one particular solution x_0 of (2.22c) as

$$x_0 = A_{eq}^{\dagger} b_{eq}, \tag{2.23}$$

where $A_{eq}^{\dagger}$ is the left pseudo-inverse of matrix A_{eq}. Note that $A_{eq} \in \mathbb{R}^{p \times n}$, with $p \leq n$, i.e. the number of equality constraints have to be smaller as the number of optimized variables, since if $p = n$ then solution would be a singleton (and there is nothing to optimize). As A_{eq} might not be a squared matrix, we have to apply the pseudo inverse. Next we will find the correct direction F to the optimum via computing null space of A_{eq}

$$A_{eq} F = 0. \tag{2.24}$$

Now we can parametrize the optimized variable x with expression

$$x = x_0 + F\lambda \tag{2.25}$$

and by substituting (2.25) into (2.22) we obtain a new optimization problem

$$\lambda^{\star} = \arg\min_{\lambda} \tfrac{1}{2}\lambda^T \widetilde{H}\lambda + \widetilde{q}^T\lambda + \widetilde{c} \tag{2.26a}$$

$$\text{s.t. } \widetilde{A}\lambda \leq \widetilde{b}, \tag{2.26b}$$

where $\widetilde{H} = F^T H F$, $\widetilde{q} = F^T q + F^T H x_0$, $\widetilde{c} = \frac{1}{2} x_0^T H x_0 + q^T x_0 + c$, $\widetilde{A} = AF$ and $\widetilde{b} = b - Ax_0$. Note that we have get rid of equality constraints (2.22c) as they become redundant. Finally, optimizers $x^{\star}$ can be derived by solving (2.26) and plugin optimal distance $\lambda^{\star}$ back into (2.25) as

$$x^{\star} = x_0 + F\lambda^{\star}. \tag{2.27}$$

For better interpretation, this transformation is depicted in Figure 2.9. Here, ellipsoids represent contours of the objective function (2.22a) and black-dashed line equality constraints (2.22c). Inequality constraints (2.22b) are omitted for brevity. Firstly, we have computed a feasible point x_0 that lies in the straight line, which represents equality constraints. And as we know also the optimizer $x^{\star}$ has to be situated on this line. Therefore, we have computed the direction F which can slide the feasible point $x^{\star}$ to the optimum along the line. The only missing piece is the length of the step which we have to proceed in order to get to this optimum. And this distance is exactly denoted by λ that is optimized in the modified optimization problem (2.26). Subsequently, the original optimizer is then easily obtained via (2.27).

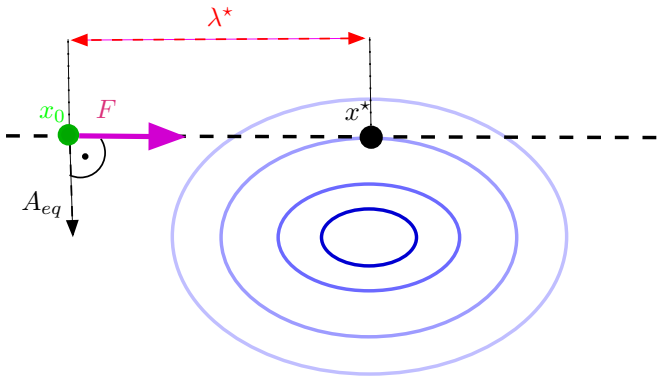


Figure 2.9: Figure illustrates parametrization technique used to eliminate equality constraints from the optimization problem.

Solution Properties of QP

Let us denote $\mathcal{P}$ as a polyhedron constituted by the inequalities in (2.21b), $J^\star$ the optimal cost and $x^\star$ a solution of (QP). Here we can encounter three scenarios:

Case 1: The problem is bounded, $J^\star > -\infty$, and the solution $x^\star$ lies strictly in the interior of polyhedron $\mathcal{P}$.

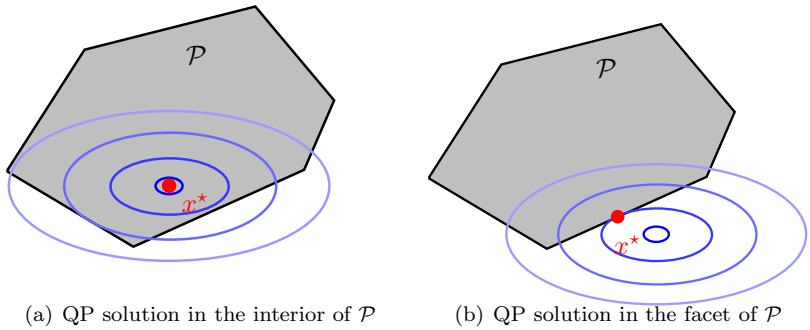
Case 2: The problem is bounded, $J^\star > -\infty$, and the solution $x^\star$ lies on the boundary of polyhedron $\mathcal{P}$.

Case 3: The problem is infeasible, $J^\star = \infty$, caused by an empty polyhedron $\mathcal{P}$.

Quadratic programming has a specific property, compared to LP, which is that the solution $x^\star$ is always unique. This fact can be seen in Figure 2.10, where first two scenarios are depicted.

2.3 Non-convex Optimization Problems

As we have already pointed out, convex optimization problems are much easier to solve and thus are more preferable. The downside, however, is that sometimes we can not formulate our problem as a convex optimization problem, e.g. due to its natural (non-convex) behavior. As an example we can mention transmissions (gearboxes) or on-off valves, which can operate only with finite number of levels.



(a) QP solution in the interior of $\mathcal{P}$ (b) QP solution in the facet of $\mathcal{P}$

Figure 2.10: Two different QP solutions are depicted here. First one 2.10(a) lies in the interior of the polytope $\mathcal{P}$, while the other one strictly on the $\mathcal{P}$ facet. This is the consequence of two distinguish objective functions, which are here denoted by blue ellipsoids, while the optimizer is marked as the red dot.

Therefore, with these types of systems, we usually end up in non-convex optimization problems where objective function or (and) constraints are non-convex.

From this category we will be particularly interested in a specific type of problem which is called Mixed Integer Programming (MIP). These optimization problems are characterized by containing both continuous variables and integer variables, i.e. a new domain of optimization problems is

$$\mathcal{X} \subseteq \{[x, \delta] \mid x \in \mathbb{R}^n, \delta \in \mathbb{N}^{n_q}\}. \tag{2.28}$$

Subsequently, a general formulation of a MIP can be stated as

$$J^\star = \inf_{x,\delta} f_0(x, \delta) \tag{2.29a}$$

$$\text{s.t. } [x, \delta] \in \mathcal{S} \subseteq \mathcal{X}. \tag{2.29b}$$

Obviously, a particular category of MIP is when integer variables $\delta \in \{0, 1\}$, i.e. are binary variables. In the sequel, we complete our overview by two non-convex functions, that will be used in this thesis, and two standard formulations of MIP. Deeper insight into this type of programming can be find e.g. in (Bemporad and Morari, 1999; Borrelli et al., 2016; Wolsey, 1998).

2.3.1 Non-convex functions

Here, two (generally) non-convex functions are presented, each of which is defined over polytopic set described in Definition 2.2.12.

Definition 2.3.1 (Polytopic piecewise affine (PWA) function) *Function $f : \Omega \to \mathbb{R}^m$ is called a piecewise affine (PWA) function over polytopes if*

1. *$\Omega \subset \mathbb{R}^n$ is a polytope,*
2. *there exist polytopes $\mathcal{R}_i$, $i = 1, \dots, M$ such that $\{\mathcal{R}_i\}_{i=1}^M$ is the partition of Ω,*
3. *for each $i = 1, \dots, M$ we have $f(x) = F_i x + g_i$, with $F_i \in \mathbb{R}^{m \times n}$, $g_i \in \mathbb{R}^m$.*

Definition 2.3.2 (Polytopic piecewise quadratic (PWQ) function) *We call function $f : \Omega \to \mathbb{R}$ a piecewise quadratic (PWQ) function over polytopes if*

1. *$\Omega \subset \mathbb{R}^n$ is a polytope,*
2. *there exist polytopes $\mathcal{R}_i$, $i = 1, \dots, M$ such that $\{\mathcal{R}_i\}_{i=1}^M$ is the partition of Ω,*
3. *for each $i = 1, \dots, M$ we have $f(x) = x^T H_i x + F_i x + g_i$, with $H_i \in \mathbb{R}^{n \times n}$, $F_i \in \mathbb{R}^n$, $g_i \in \mathbb{R}$.*

2.3.2 Mixed Integer Linear Programming

A non-convex MIP (2.29) with linear objective function (2.29a) and linear constraints (2.29b) is called Mixed Integer Linear Program (MILP) that can be generally formulated as

$$J^\star = \min_{x,\delta} c^T x + d^t \delta \tag{2.30a}$$

$$\text{s.t. } Ax + B\delta \leq b, \tag{2.30b}$$

$$A_{eq} x + B_{eq} \delta = b_{eq}, \tag{2.30c}$$

with matrices $c \in \mathbb{R}^n$, $d \in \mathbb{R}^{n_q}$, $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{m \times n_q}$, $b \in \mathbb{R}^m$,$A_{eq} \in \mathbb{R}^{p \times n}$, $B_{eq} \in \mathbb{R}^{p \times n_q}$, $b_{eq} \in \mathbb{R}^p$, decision variables $x \in \mathbb{R}^n$ and $\delta \in \mathbb{N}^{n_q}$. Here is evident, that the only structural difference between standard LP (2.18) and MILP (2.30) formulation is in the additional (integer) type of optimization variables δ.

2.3.3 Mixed Integer Quadratic Programming

Non-convex MIP (2.29) with Quadratic objective function (2.29a) and linear constraints (2.29b) is called Mixed Integer Quadratic Program (MIQP) that can be generally stated as

$$J^\star = \min_{x,\delta} \tfrac{1}{2}x^T H_1 x + \tfrac{1}{2}x^T H_2 \delta + \tfrac{1}{2}\delta^T H_3 \delta + q_1^T x + q_2^T \delta + c \tag{2.31a}$$

$$\text{s.t. } Ax + B\delta \leq b, \tag{2.31b}$$

$$A_{eq}x + B_{eq}\delta = b_{eq}, \tag{2.31c}$$

with matrices $H_1 \in \mathbb{R}^{n\times n}$, $H_2 \in \mathbb{R}^{n\times n_q}$, $H_3 \in \mathbb{R}^{n_q\times n_q}$, $q_1 \in \mathbb{R}^n$, $q_2 \in \mathbb{R}^{n_q}$, $c \in \mathbb{R}$, $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{m\times n_q}$, $b \in \mathbb{R}^m$, $A_{eq} \in \mathbb{R}^{p\times n}$, $B_{eq} \in \mathbb{R}^{p\times n_q}$, $b_{eq} \in \mathbb{R}^p$, decision variables $x \in \mathbb{R}^n$ and $\delta \in \mathbb{N}^{n_q}$. Similarly, as in the comparison between LP and MILP, the only structural difference between QP and MIQP is the presence of integer optimization variables δ.

2.3.4 Aspects and Solutions of MIP

In brevity, solving a MILP or MIQP can be represented as seeking for such optimizer $[x^\star, \delta^\star]$ where the objective function $J^\star$ reaches its optimum, with respect to all constraints, and variables $\delta^\star$ attain integer values. From the formulation perspective MILP and MIQP do not distinguish much from their subgroups LP and QP. However, the main difference between them can be found in the techniques, used to solve these sort of problems, and the computational burden associated with them.

One of the most straightforward way how one can solve such problems is to omit restriction that δ is a vector of integer variables and solve the corresponding LP/QP (relaxed) problem. Subsequently, optimization variables δ are rounded to the nearest integer value. The problem, however, is that the solution of such approach may be not optimal or even not feasible. This issue is depicted in Figure 2.11. Here, it is evident that by rounding the relaxed solution $x^\star_{rel}$ we would not get to the optimum $x^\star_{MIP}$, but outside of the MIP feasible domain. Therefore we can say that the accuracy is the price that we need to pay for the reduced computational complexity in this approach.

Complexity of MIP can be also interpreted on one of the most intensively studied optimization problems named Traveling Salesman Problem (TSP) (see e.g. Ap-

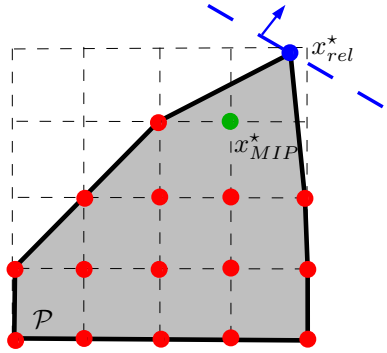


Figure 2.11: Figure shows an elementary MIP technique. $\mathcal{P}$ denotes feasible set, red dots represents feasible integer values and dashed blue line is objective function specified direction vector. Finally, the blue dot is solution of relaxed problem and the green dot solution of integer-valued problem.

plegate et al. (2006)). Popularity of this problem is established due to its wide applicability mainly in planing and logistics (see e.g. Klaučo et al. (2014); Oravec et al. (2015)). Here the main idea is to find the shortest path that intersects all given places. The most straightforward way how to solve such problem is to sequentially examine all possible transitions, but obviously the problem becomes quickly unpractical with the increasing number of integer variables.

Nowadays there is a wide range of sophisticated methods that decrease the number of investigated integer combinations and thus provide better complexity scaling for MIP. The commonly used are branch-and-bound or branch-and-cut methods (see e.g. Linderoth and Ralphs (2004)). Furthermore improvement has been also detected in numerous solvers e.g. (Gurobi Optimization, 2014; ILOG, Inc., 2003).

Chapter 3

Model Predictive Control

Model Predictive Control (MPC) is a control strategy which belongs to a class of optimal control algorithms. It exploits a mathematical model of the process to predict dynamics of the plant over a certain prediction horizon. Subsequently, an optimization technique is employed to compute adequate control actions that lead to a desired control performance.

This chapter provides a basic introduction to MPC with a brief history and comparison of properties and characteristics. Furthermore, it defines and discusses several standard mathematical formulations of MPC.

3.1 Introduction to MPC

3.1.1 Origins of MPC

The history of MPC can be traced back to late 1970's with the outcome of two pioneering works. The first one was called Model Predictive Heuristic Control (MPHC) shown by Richalet et al. (1976, 1978) along with its solution software IDCOM, which stands for Identification and Command. The second one was referred as Dynamic Matrix Control (DMC) that was introduced in 1979 by Shell Oil engineers (Cutler and Ramaker, 1979, 1980). Both of these approaches belong to the so-called *first generation* of MPC and were developed by engineers in the industry, who benefited from the rapid development of computer technology. This dramatic change had a great impact on industrial process control and set a new foundation

to industrial MPC paradigm. From this point forward, many researchers have put a lot of effort to further polish this methodology (e.g. establishing stability, that was firstly incorporated only via proper tuning, embedding constraints, increasing flexibility, etc.) and they are busy with this task ever since. For interested readers, an extensive historical survey of industrial MPC can be find e.g. in Qin and Badgwell (2003).

3.1.2 MPC Nowadays

During the last decades, MPC has experienced a significant change from the theoretical research to practical applications. At first, it has been applied only on systems with a slow dynamics, which was caused mainly due to computational burden. But nowadays, with the technological progress and many accelerating solvers (e.g. Gurobi Optimization (2014); ILOG, Inc. (2003)) or toolboxes (e.g. Houska et al. (2011); Wächter and Biegler (2006)), it is a widely adopted control strategy especially in the process control, see, e.g., Maciejowski (2001), Qin and Badgewell (2003). Its success stems from its natural capability of designing controllers for multidimensional systems while taking into account all of the systems physical constraints and performance specifications which are explicitly embedded in the optimization problem. Based on these advantages model predictive control can not only reduce the material and energy consumption, but also to minimize the negative impacts on the environment, while maximizing the profit. And it is these economical and environmental aspects which, nowadays, are attracting increasing attention in the design of advanced control algorithms.

Needless to say, MPC is one of the most modern control policy that overcomes the shortcomings of conventional control approaches, such as Proportional-Integral-Derivative controller (PID) or Linear-Quadratic Regulator (LQR). A basic comparison between these methodologies is depicted in Table 3.1. Here, the superiority of MPC properties are evident. However it should be noted that other commonly used control algorithms, even thought that they have limited applicability, can still find their place and utilization. E.g. the most widespread controllers are PIDs that due to their cheap and effective implementation, mainly on single-input single-output (SISO) systems, cover approximately 90% of control loops in industry.

On the other hand, the consequence of these embedded constraints and the finite prediction of systems dynamics leads to a complex optimization problem,

	PID	LQR	MPC
Model	Linear	Linear	(Non-)Linear, Hybrid
Dimension	SISO	MIMO	MIMO
Time variance	Time-invariant	Time-invariant	Time-variant
Optimization	NO	YES	YES
Constraints	NO	NO	YES
Implementation	Cheap, Easy	Cheap, Easy	Expensive, Complex

Table 3.1: Comparation of three commonly used control approaches.

which (in order to guarantee optimality, stability and constraints satisfaction) has to be solved within duration of one sample instant. Therefore, such an optimization requires a lot of computation power and last, but not least an appropriate solver. However, if such limitations are not meet in the control hardware, then one need to change the control setup e.g. to simplify the model of the process (what will decrease the control performance), use open loop implementation (if it is possible), or apply a different control method (e.g. explicit MPC).

3.1.3 Receding Horizon Control

MPC is usually implemented in a closed-loop fashion in an approach referred to as the Receding Horizon Control (RHC). The basic principle of RHC, illustrated in Figure 3.1, is to solve an MPC optimization problem to obtain a sequence of optimal control actions, over a finite horizon, from which only the first element of this sequence is taken and applied to the process to achieve a feedback. The remaining control inputs are simply discarded. At the next sampling instant, a new state measurement is obtained, the prediction horizon is shifted and then the optimization problem is solved once again. This yields a new sequence of optimal control inputs, from which the first action is extracted and executed in the controlled process. Subsequently, the whole procedure then repeats ad-infinitum. Generally, implementation of MPC in a RHC fashion can be summarized as follows:

1. Derive a mathematical model of the controlled plant.

2. Formulate an optimization problem.
3. At a time instant t acquire state measurements from the plant, i.e. $x(t)$.
4. Solve the optimization problem which yields a sequence of optimal control actions along the prediction horizon N, i.e. $U^{\star}_{\text{ol}} = [u_0^{\star T}, u_1^{\star T}, \ldots, u_{N-1}^{\star T}]^T$.
5. Apply only the first element of this sequence to the plant, i.e. $u_0^{\star T}$.
6. Repeat this procedure from the third step as the time progress to the next sampling period T_s, i.e. $t + T_s$.

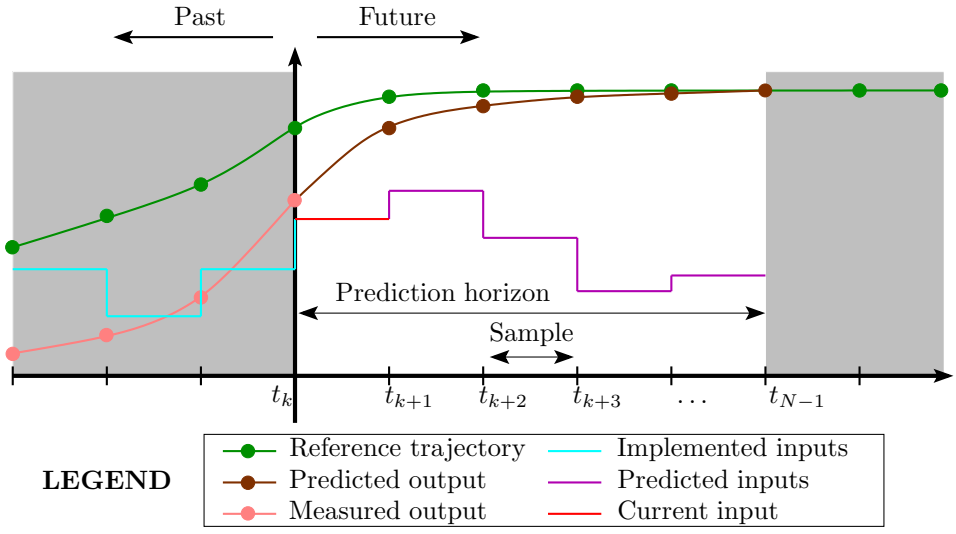


Figure 3.1: Characteristic scheme of MPC policy.

3.1.4 Open-loop Implementation

MPC can be implemented also in open-loop manner, where the entire optimal control sequence $U^{\star}_{\text{ol}}(t)$ is implemented. This means that the optimization problem has to be solved not at each sampling instant T_s as in RHC, but only when all optimal control actions were applied, i.e. each NT_s steps.

This implementation technique might, however, lead to some unwanted consequences. One of them is depicted in Figure 3.2. Here, three different control trajectories are shown. The black one illustrates the real state trajectory of the controlled system, while other denote predicted state trajectories with accurate model

(green line) and inaccurate model (red line). It can be seen that even thought that all originates from state measurement $x(t)$, their control performances distinguish even more with each predicted step. It follows directly from this situation that the open-loop implementations with weakly design models exhibit pure control performances which might lead to hazardous scenarios. On the other hand, in the RHC, this problem is mitigated (yet not directly eliminated) with continuously updating state measurements $x(t)$.

Moreover, it should be noted that under certain conditions both (closed-loop and open-loop) implementations are identical see e.g. Mayne et al. (2000). This phenomenon will be closely discussed and exploited later.

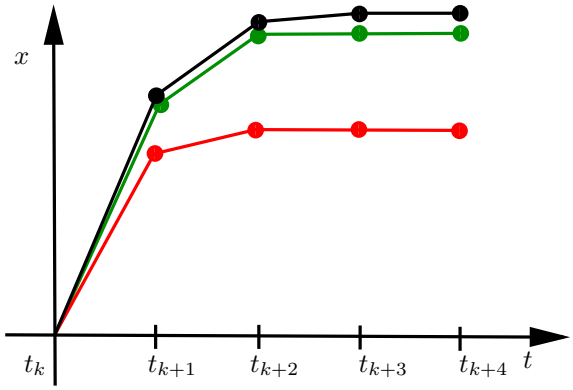


Figure 3.2: Plant-model mismatch in open-loop implementation.

3.2 Basic Mathematical Formulations of MPC

As we have shown in the previous section, MPC policy is well adopted control strategy which allows one to adjust its formulation to the need of the controlled system. Particularly, this strategy can handle and incorporate wide variety of constraints, penalization expressions or models, each of which might lead to a different type of optimization problem. Therefore the consequence of this flexibility is that there exists a rich diversity of MPC formulations.

In this section we will just scratch the surface of this topic and provide only a basic overview of standard MPC formulations which are commonly used in process control.

3.2.1 General MPC Formulation

Consider prediction model in discrete-time, which approximates the controlled process, given as

$$x_{k+1} = f(x_k, u_k), \tag{3.1}$$

with vector of states $x_k \in \mathbb{R}^n$ and vector of inputs $u_k \in \mathbb{R}^m$. Further, let the prediction model (3.1) be subject to constraints

$$x_k \in \mathcal{X},\ u_k \in \mathcal{U}, \tag{3.2}$$

where $\mathcal{X} \subseteq \mathbb{R}^n$, $\mathcal{U} \subseteq \mathbb{R}^m$. Then a general form of the MPC optimization problem can be formulated as

$$J^\star = \min_{u_0,\dots,u_{N-1}} \ell(x_N) + \sum_{k=0}^{N-1} \ell(x_k, u_k) \tag{3.3a}$$

$$\text{s.t. } x_{k+1} = f(x_k, u_k),\ k = 0, \dots, N-1, \tag{3.3b}$$

$$x_k \in \mathcal{X},\ k = 0, \dots, N, \tag{3.3c}$$

$$u_k \in \mathcal{U},\ k = 0, \dots, N-1, \tag{3.3d}$$

$$x_0 = x(t), \tag{3.3e}$$

where x_k, u_k denote values of state and inputs predicted at the k-th stage of the prediction horizon $N \in \mathbb{N}$. These predictions are made based on the mathematical model (3.3b) and initialized from the state measurement $x(t)$ in (3.3e). Next, the term $\ell(\cdot)$ in (3.3a) refers to a stage cost that penalizes the predicted variables x_k, u_k and $\mathcal{X}$,$\mathcal{U}$ are sets representing state (3.3c) and input (3.3d) constraints.

Finally, the solution of the optimization problem (3.3) yields a vector of optimal open-loop control actions $U^\star_{\text{ol}} = [{u^\star_0}^T, \dots, {u^\star_{N-1}}^T]^T$, where the objective function (3.3a) reaches its optimum $J^\star$. Moreover, since in the RHC implementation we are interested only in the first element of the vector $U^\star_{\text{ol}}$, we can then define the RHC feedback law $\mu : \mathbb{R}^n \to \mathbb{R}^m$ as

$$\kappa(x(t)) = [\mathbf{I}_{m\times m}, \mathbf{0}_{m\times m}, \dots, \mathbf{0}_{m\times m}] U^\star_{\text{ol}}(x(t)). \tag{3.4}$$

As was mentioned, complexity of the optimization problem (3.3), as well as its computational burden, may differ based on the type of each of its component (e.g. objective function, constraints or decision variables). Next we introduce only standard techniques, to formulate these components, that lead to convex optimization problems.

3.2.2 Prediction Model

The prediction model acts as the corner stone in MPC policy which is used to predict the behavior of the controlled system. Subsequently, based on these predictions, optimal control actions are being computed. In practice we often encounter with processes, whose behavior is highly nonlinear (e.g. neutralization process) and described e.g. by ordinary differential equations (ODE). But in control theory, for simplicity, a more preferable state-space model is being used, which however represents only an approximation of a real system that behaves in a more complicated nonlinear fashion. Therefore in this section we firstly state a linear time-invariant (LTI) model, which is used in the majority of this thesis, and then we show how one can derive such model.

In this thesis we consider discrete-time LTI systems in a state-space form, whose dynamics can be described by

$$x(t+1) = Ax(t) + Bu(t), \tag{3.5}$$

with state vector $x(t) \in \mathbb{R}^n$, input vector $u(t) \in \mathbb{R}^m$, and with matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, where the pair (A, B) is controllable. Next, let the system (3.5) be subjected to constraints

$$x(t) \in \mathcal{X}, \tag{3.6a}$$

$$u(t) \in \mathcal{U}, \tag{3.6b}$$

where $\mathcal{X} \in \mathbb{R}^n, \mathcal{U} \in \mathbb{R}^m$ are polytopes that contain origin in their interiors.

Remark 3.2.1 *Note that we keep the system in* (3.5) *quite simple by neglecting disturbances, model uncertainties and the output equation. Moreover, we assume to have knowledge of the full state measurements. It will be explicitly specified if a different setup will be considered.*

Linearization

This section describes a commonly used linearization technique used to transform nonlinear model of the system into linear one. Even thought that thesis assumes model in (3.5), here we provide more general approach.

Given is a nonlinear system which is described by following dynamics[1]

$$\frac{d\mathfrak{x}}{dt} = f(\mathfrak{x}, \mathfrak{u}), \tag{3.7a}$$

$$\mathfrak{y} = g(\mathfrak{x}, \mathfrak{u}), \tag{3.7b}$$

where $\mathfrak{x} \in \mathbb{R}^n$ is a state vector, $\mathfrak{u} \in \mathbb{R}^m$ is an input vector and $\mathfrak{y} \in \mathbb{R}^r$ denotes output vector. Consider an operating point $\mathfrak{x}_0$, associated with input $\mathfrak{u} = \mathfrak{u}_0$ and output $\mathfrak{y} = \mathfrak{y}_0 = g(\mathfrak{x}_0, \mathfrak{u}_0)$, where the system (3.7) is going to be controlled (or in its neighborhood). Then the linear continuous-time systems, can by derived by linearization around $\mathfrak{x}_0$ via Taylors expansion as

$$\frac{d\mathfrak{x}}{dt} = f(\mathfrak{x}, \mathfrak{u}) \approx f(\mathfrak{x}_0, \mathfrak{u}_0) + \left.\frac{\partial f(\mathfrak{x}, \mathfrak{u})}{\partial \mathfrak{x}}\right|_{\mathfrak{x}_0, \mathfrak{u}_0} x(t) + \left.\frac{\partial f(\mathfrak{x}, \mathfrak{u})}{\partial \mathfrak{u}}\right|_{\mathfrak{x}_0, \mathfrak{u}_0} u(t), \tag{3.8a}$$

$$\mathfrak{y} = g(\mathfrak{x}, \mathfrak{u}) \approx g(\mathfrak{x}_0, \mathfrak{u}_0) + \left.\frac{\partial g(\mathfrak{x}, \mathfrak{u})}{\partial \mathfrak{x}}\right|_{\mathfrak{x}_0, \mathfrak{u}_0} x(t) + \left.\frac{\partial g(\mathfrak{x}, \mathfrak{u})}{\partial \mathfrak{u}}\right|_{\mathfrak{x}_0, \mathfrak{u}_0} u(t), \tag{3.8b}$$

with Jacobian matrices[2] $\frac{\partial f(\mathfrak{x},\mathfrak{u})}{\partial \mathfrak{x}}$, $\frac{\partial f(\mathfrak{x},\mathfrak{u})}{\partial \mathfrak{u}}$, $\frac{\partial g(\mathfrak{x})}{\partial \mathfrak{x}}$, $\frac{\partial g(\mathfrak{x},\mathfrak{u})}{\partial \mathfrak{u}}$ and where $x(t) = \mathfrak{x} - \mathfrak{x}_0$, $u(t) = \mathfrak{u} - \mathfrak{u}_0$ denote deviation variables. Next, as $\mathfrak{x}_0$ is only a particular value of state, we have that

$$\frac{dx(t)}{dt} = \hat{A}x(t) + \hat{B}u(t) + f(\mathfrak{x}_0, \mathfrak{u}_0), \tag{3.9a}$$

$$y(t) = \hat{C}x(t) + \hat{D}u(t), \tag{3.9b}$$

with an output in its deviation form $y(t) = \mathfrak{y} - \mathfrak{y}_0$ and matrices $\hat{A}$, $\hat{B}$, $\hat{C}$ and $\hat{D}$ denote appropriate Jacobian matrices. Furthermore, it is a standard practice to choose $(\mathfrak{x}_0, \mathfrak{u}_0)$ as an equilibrium point of the system, what reduces (3.9) and we obtain a LTI state-space system, in a continuous domain, defined as

$$\frac{dx(t)}{dt} = \hat{A}x(t) + \hat{B}u(t), \tag{3.10a}$$

$$y(t) = \hat{C}x(t) + \hat{D}u(t), \tag{3.10b}$$

Finally, by applying a discretization technique (e.g. Euler's method, Zero-order hold, etc.), LTI system in (3.10) can be defined in its discrete-time domain as

$$x(t+1) = Ax(t) + Bu(t), \tag{3.11a}$$

$$y(t) = Cx(t) + Du(t), \tag{3.11b}$$

[1] For simplicity, time domain, e.g. $\mathfrak{x}(\mathfrak{t}) = \mathfrak{x}$, is omitted

[2] Here, terms of a higher-order in x and u are neglected.

with state vector $x(t) \in \mathbb{R}^n$, input vector $u(t) \in \mathbb{R}^m$, output vector $y(t) \in \mathbb{R}^r$, and with matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{r\times n}$ and $D \in \mathbb{R}^{r\times m}$, where a constant interval between each input sampling $T_s = \triangle t$ is assumed.

Remark 3.2.2 *Generally, there are three classes of modeling techniques. The white-box approach, also known as a-priori modeling, exploits physical and chemical laws to devise the model. The problem from this direction, however, is that such constructed models are usually too complex, and to tackle this issue the aforementioned linearization is employed. On the other hand, if there are no mathematical descriptions of a system dynamics in hand (e.g. as in* (3.7)*), one can consider the black-box approach, also called a-posteriori modeling. Here, the model is constructed via experimental data which describe the relationship between input and output signals of the system. Nowadays, since it is the most common technique, there exist many identification methods which are embedded in numerous toolboxes (Oravec and Bakošová, 2012; Overschee and Moor, 1994). Finally, the grey-box approach combines both previous techniques, i.e. it exploits insight into theory and experimental data analysis.*

Remark 3.2.3 *It should be noted that it is standard practice to control the system only in the operating point, or in its neighborhood, where such linearized models are satisfactory. However, for the systems with strong nonlinearity, or with increasing control range, this models can be imprecise and thus might lead to pure control performance. In this case a nonlinear dynamic has to be considered.*

3.2.3 Objective Function

Objective function, in a concept of MPC, is a mathematical term that assigns a specific cost to each predicted variable to provide desired control performance, or shortly it defines the aim of the control. It is a standard practice to define objective function as

$$J = \|Q_x x_N\|_p + \sum_{k=0}^{N-1} (\|Q_x x_k\|_p + \|Q_u u_k\|_p) \tag{3.12}$$

where p denotes a norm and $Q_x \in \mathbb{R}^{n\times n}$, $Q_u \in \mathbb{R}^{m\times m}$ are weighting matrices, which assign costs to vector of states and inputs, respectively. Since these matrices are one of the most common subjects in the tuning procedure of MPC policy, thus we will next aim at norms and discuss their aspects in the concept of MPC.

A norm is a convex function (see Section 2.2.1) that assigns strictly positive length of all nonzero vectors, i.e. it specifies the length or size of a vector. Formulation of a general p-norm of a vector $x \in \mathbb{R}^n$, or shortly $\|x\|_p$, can be given as

$$\|x\|_p = \left(\sum_i |x|^p \right)^{1/p} \tag{3.13}$$

with a following properties:

- $\|x\|_p \geq 0$,
- $\|x\|_p = 0$, if and only if $x = 0$,
- $\|cx\|_p = |c|\|x\|_p$, for any $c \in \mathbb{R}$,
- $\|x_1 + x_2\|_p \leq \|x_1\|_p + \|x_2\|_p$.

There is a wide variety of norms (as shown e.g. in (Gradshteyn and Ryzhik, 2007, p. 1081)), but the commonly used norms are Manhattan (1-norm), Euclidean (2-norm) or Maximum (∞-norm) norms.

Manhattan norm

Manhattan norm, also referred to as the 1-norm, is defined as a sum of absolute values of all vector elements x_i, i.e.

$$\|x\|_1 = \sum_i |x_i|. \tag{3.14}$$

Consider now a following example. Given is an optimization problem as in (2.17), where the length of the vector $x \in \mathbb{R}^n$ is going to be minimized

$$J^\star = \min_x \|Cx\|_1 \tag{3.15a}$$

$$\text{s.t. } Ax \leq b, \tag{3.15b}$$

where $C = \text{diag}(c)$, $c \in \mathbb{R}^n$. The problem (3.15) is no longer a LP, since the objective function (3.15a), depicted in Figure 3.3(a) for an arbitrary element x_i, has form of a piecewise linear function (PWL). To avoid any numerical difficulties, it is convenient to perform a transformation of (3.15) into a standard form of LP

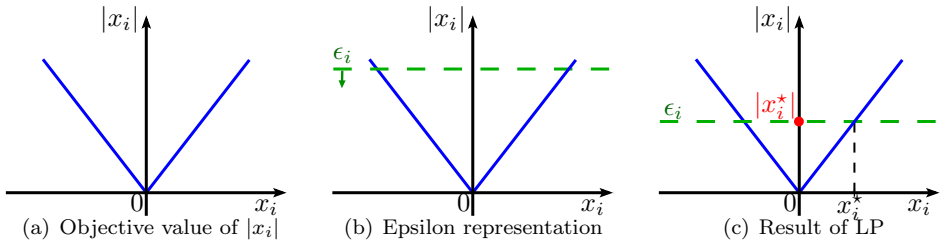
as in (2.17). This can be achieved be introducing an epigraph formulation

$$J^\star = \min_{x,\epsilon} \sum_i \epsilon_i \tag{3.16a}$$

$$\text{s.t. } Ax \leq b, \tag{3.16b}$$

$$-\epsilon \leq Cx \leq \epsilon, \tag{3.16c}$$

where the sum of all epigraphs $\epsilon \in \mathbb{R}^n$ is minimized (Figure 3.3(b)). The optimization is terminated, when $x \in \mathbb{R}^n$ meets the added constraints (3.16c). Subsequently, a solution $x^\star \in \mathbb{R}^n$ is obtained (Figure 3.3(c)). It is important to note, that in the transformed form (3.16), the number of optimization variables is maintained, but number of constraints is increased from m to $(m + 2n)$.



(a) Objective value of $|x_i|$ (b) Epsilon representation (c) Result of LP

Figure 3.3: Figures show the transformation of basic MPC optimization defined as PWL problem 3.3(a), due to the Manhattan norm, which is transformed into LP (3.16) via epigraph formulation 3.3(b). The result of such optimization problem is depicted in 3.3(c). The green-dashed line represents the epigraph function ϵ_i that is minimized, what is indicated by the green arrow. The solution of this optimization $|x_i^\star|$ is illustrated by the red dot, through which the state $x_i^\star$ is determined.

Maximum Norm

Maximum norm, also referred as ∞-norm, is defined as a maximum among all absolute values of vector x, i.e.

$$\|x\|_\infty = \max_i |x_i|. \tag{3.17}$$

Similarly as in Manhattan norm, by substituting the 1-norm by the ∞-norm in (3.15), one obtains the objective function (3.17) in form of a PWL function.

To tackle this problem, one can exploit the epigraph formulation, what leads to

$$J^\star = \min_{x,\epsilon} \epsilon \tag{3.18a}$$

$$\text{s.t. } Ax \leq b, \tag{3.18b}$$

$$-\mathbf{1}_n\epsilon \leq Cx \leq \mathbf{1}_n\epsilon, \tag{3.18c}$$

where only the worst absolute value among vector x is minimized $\epsilon \in \mathbb{R}$. Hence, the number of constraints is here increased from m to $m + 2$.

Euclidean Norm

Euclidean norm can be interpreted as the shortest distance in the euclidean space and is defined by

$$\|x\|_2 = \sqrt{\sum_i x_i^2}. \tag{3.19}$$

By plugging (3.19) into the optimization problem (2.17), to penalize the distance of the vector x (from the origin), leads to neither LP nor QP (due to the square root). This obstacle can be overcome by squaring the norm, i.e. $\|x\|_2 \implies \|x\|_2^2$. The optimization problem (3.15) will be then a QP in a following form

$$J^\star = \min_x x^T Cx \tag{3.20a}$$

$$\text{s.t. } Ax \leq b. \tag{3.20b}$$

One should note that the consequence of this transformation into QP is that, compared to the expression $\sqrt{x^T Cx}$, in the interval $[-1, 1]$ cost $J^\star$ is undervalued and in the interval $(-\infty, -1) \cup (1, \infty)$ cost $J^\star$ is overvalued. But, most importantly, the optimizer $x^\star$ is identical to the optimizer which minimizes $\sqrt{x^T Cx}$.

Remark 3.2.4 *Aforementioned examples demonstrate the Manhattan, Maximum and Euclidean norms only for one vector x. In MPC policies, e.g. in* (3.12)*, one needs to apply a norm for each predicted state $\|x_k\|_p$ and input $\|u_k\|_p$ vector. We need to keep in mind that in the case of* 1*-norm or ∞-norm this increases the number of variables ϵ, thus the number of constrains.*

3.2.4 MPC as a Linear Program

Consider system (3.5) to be used as a prediction model and subjected to polyhedral constraints

$$\mathcal{X} = \{x \in \mathbb{R}^n \mid H_x x \leq K_x\}, \tag{3.21a}$$

$$\mathcal{U} = \{u \in \mathbb{R}^m \mid H_u u \leq K_u\}, \tag{3.21b}$$

with $H_x \in \mathbb{R}^{n_x \times n}$, $K_x \in \mathbb{R}^{n_x}$, $H_u \in \mathbb{R}^{n_u \times m}$ and $K_u \in \mathbb{R}^{n_u}$, where n_x, n_u is the number of half-spaces corresponding to state and input polyhedron respectively. Next, let 1-norm (3.14) or ∞-norm (3.17) be used in the objective function in (3.12). Then the MPC problem (3.3) can be stated as a linear optimization problem of a form

$$J^\star = \min_{u_0,\ldots,u_{N-1},\epsilon^x,\epsilon^u} \mathbf{1}_n^T \epsilon_N^x + \sum_{k=0}^{N-1} (\mathbf{1}_n^T \epsilon_k^x + \mathbf{1}_m^T \epsilon_k^u) \tag{3.22a}$$

$$\text{s.t. } x_{k+1} = Ax_k + Bu_k,\ k = 0,\ldots,N-1, \tag{3.22b}$$

$$x_0 = x(t), \tag{3.22c}$$

$$H_x x_k \leq K_x,\ k = 0,\ldots,N, \tag{3.22d}$$

$$H_u u_k \leq K_u,\ k = 0,\ldots,N, \tag{3.22e}$$

$$-\Gamma_n \epsilon_k^x \leq Q_x x_k,\ k = 0,\ldots,N, \tag{3.22f}$$

$$-\Gamma_n \epsilon_k^x \leq -Q_x x_k,\ k = 0,\ldots,N, \tag{3.22g}$$

$$-\Gamma_m \epsilon_k^u \leq Q_u u_k,\ k = 0,\ldots,N-1, \tag{3.22h}$$

$$-\Gamma_m \epsilon_k^u \leq -Q_u u_k,\ k = 0,\ldots,N-1, \tag{3.22i}$$

where $x_k \in \mathbb{R}^n$, $u_k \in \mathbb{R}^m$ are state and input vectors at the k-th prediction instance, $x(t) \in \mathbb{R}^n$ is a state measurement at time t, $N \in \mathbb{N}$ is a prediction horizon and $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$ are model matrices. Next, in 1-norm Γ_j denotes an identity matrix $\Gamma_j = \mathbf{I}_j$ and if ∞-norm is applied then $\Gamma_j = \mathbf{1}_j$, i.e. $\Gamma_j = [1,\ldots,1]^T \in \mathbb{R}^j$. Moreover, epigraphs at k-th prediction stage are denoted in 1-norm by vectors $\epsilon_k^x \in \mathbb{R}^n$, $\epsilon_k^u \in \mathbb{R}^m$ and in case of ∞-norm by vectors $\epsilon_k^x \in \mathbb{R}$, $\epsilon_k^u \in \mathbb{R}$.

Remark 3.2.5 *It should be noted that even though both norms admit an LP representation, the two formulations have their own disadvantages, e.g. while ∞-norm might lead to a pure control performance (as only the largest violations are suppressed), in the* 1*-norm the number of variables is increased.*

3.2.5 MPC as a Quadratic Program

Consider system (3.5) to be used as a prediction model, subjected to polyhedral constraints as in (3.21). Next, let the squared 2-norm (3.19) be used in the objective function in (3.12). Then the MPC problem (3.3) can be stated as a quadratic optimization problem of a form

$$J^\star = \min_{u_0,\dots,u_{N-1}} x_N^T Q_N x_k + \sum_{k=0}^{N-1} (x_k^T Q_x x_k + u_k^T Q_u u_k) \tag{3.23a}$$

$$\text{s.t. } x_{k+1} = Ax_k + Bu_k,\ k = 0,\dots,N-1, \tag{3.23b}$$

$$x_0 = x(t), \tag{3.23c}$$

$$H_x x_k \leq K_x,\ k = 0,\dots,N, \tag{3.23d}$$

$$H_u u_k \leq K_u,\ k = 0,\dots,N-1, \tag{3.23e}$$

where $x_k \in \mathbb{R}^n$, $u_k \in \mathbb{R}^m$ are state and input vectors at the k-th prediction instance, $x(t) \in \mathbb{R}^n$ is a state measurement at time t, $N \in \mathbb{N}$ is a prediction horizon and $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$ are model matrices.

3.2.6 Dense Formulation of QP MPC

In previous sections we have shown how one can formulate MPC policy e.g. based on used norms or control target. Yet, to obtain the desired optimal control actions, one need to employ an optimization solver that usually accepts optimization problems in their standard forms.

Generally there are two main compositions of MPC. First one is referred as *dense* formulation and second one is called *sparse* formulation. Their difference manifests from the structural composition of matrices and number of optimized variables. Particularly, *dense* formulation has lower number of optimization variables Nm compared to *sparse* formulation with $N(n+m)$ variables. However, the price which is paid for this reduction is less preferable structure of matrices, that might lead to increased computational demands. It is well known that computational burden of these formulations scales with the prediction horizon. While the *dense* formulation is preferred in a case with shorter prediction horizon, the *sparse* formulation is used for MPC problems with longer prediction horizons.

Despite the fact that there are already toolboxes (e.g. Yalmip (Löfberg, 2004)), which performs this transformation for us, for the purposes of this thesis we will

show how to transform (3.23) into its standard *dense* QP form.

To proceed let us define state and input vectors for the entire prediction horizon N as

$$X_{\text{ol}} = \begin{bmatrix} x_0 \\ \vdots \\ x_N \end{bmatrix}, U_{\text{ol}} = \begin{bmatrix} u_0 \\ \vdots \\ u_{N-1} \end{bmatrix}.$$

Then the objective function in (3.23a) can be rewritten into

$$J(x(t), U_{\text{ol}}) = X_{\text{ol}}^T \widetilde{Q}_x X_{\text{ol}} + U_{\text{ol}}^T \widetilde{Q}_u U_{\text{ol}}, \tag{3.24}$$

with matrices

$$\widetilde{Q}_x = \begin{bmatrix} Q_x & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & Q_x & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & Q_x \end{bmatrix}, \widetilde{Q}_u = \begin{bmatrix} Q_u & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & Q_u & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & Q_u \end{bmatrix},$$

which can be computed e.g. via kronecker tensor product between matrix Q_x and an identity matrix $\mathbf{I}_N$, i.e. $\widetilde{Q}_x = Q_x \otimes \mathbf{I}_{N+1}$. Next, by introducing substitution

$$x_k = A^k x(t) + \sum_{i=0}^{k-1} A^{k-i-1} B u_i \tag{3.25}$$

the state evolution based on model in (3.23b) and along the horizon N can be explicitly expressed by

$$X_{\text{ol}} = \widetilde{A} x(t) + \widetilde{B} U_{\text{ol}}, \tag{3.26}$$

with matrices $\widetilde{A} \in \mathbb{R}^{n(N+1) \times n}$, $\widetilde{B} \in \mathbb{R}^{n(N+1) \times mN}$ defined as

$$\widetilde{A} = \begin{bmatrix} \mathbf{I}_n \\ A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}, \widetilde{B} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ B & \mathbf{0} & \dots & \mathbf{0} \\ AB & B & \ddots & \vdots \\ \vdots & \vdots & \ddots & \mathbf{0} \\ A^{N-1}B & A^{N-2}B & \dots & B \end{bmatrix}. \tag{3.27}$$

By plugging the state evolution (3.26) into the objective function (3.24) and performing some straightforward mathematical transformations we get

$$J(x(t), U_{\text{ol}}) = \tfrac{1}{2} U_{\text{ol}}^T H U_{\text{ol}} + x(t)^T F U_{\text{ol}} + x(t)^T Y x(t), \tag{3.28}$$

with $H = 2\widetilde{B}^T\widetilde{Q}_x\widetilde{B} + \widetilde{Q}_u$, $F = 2\widetilde{A}^T\widetilde{Q}_x\widetilde{B}$ and $Y = \widetilde{A}^T\widetilde{Q}_x\widetilde{A}$.

With the finished objective function we can now target at state and input constraints that are assumed to be in a polyhedral form of (3.21). The state constraints in (3.23d) can by expressed for entire prediction vector N as

$$\widetilde{H}_x X_{\text{ol}} \leq \widetilde{K}_x, \tag{3.29}$$

where $\widetilde{H}_x = \mathbf{I}_{N+1} \otimes H_x$ and $\widetilde{K}_x = \mathbf{1}_{N+1} \otimes K_x$. Furthermore, by applying substitution (3.25) into (3.29) we have

$$\widetilde{H}_x\widetilde{B}U_{\text{ol}} \leq \widetilde{K}_x - \widetilde{H}_x\widetilde{A}x(t). \tag{3.30}$$

Analogically, the input constraints (3.23e), along the prediction horizon N, are denoted by

$$\widetilde{H}_u U_{\text{ol}} \leq \widetilde{K}_u, \tag{3.31}$$

with $\widetilde{H}_u = \mathbf{I}_N \otimes H_u$ and $\widetilde{K}_u = \mathbf{1}_N \otimes K_u$. By merging (3.30) with (3.31) we get

$$GU_{\text{ol}} \leq w + Ex(t), \tag{3.32}$$

with

$$G = \begin{bmatrix} \widetilde{H}_x\widetilde{B} \\ \widetilde{H}_u \end{bmatrix}, w = \begin{bmatrix} \widetilde{K}_x \\ \widetilde{K}_u \end{bmatrix}, E = \begin{bmatrix} -\widetilde{H}_x\widetilde{A} \\ \mathbf{0}_{(n_u N)\times n} \end{bmatrix}.$$

Finally, by using results from (3.28) and from (3.32), the MPC optimization problem (3.23) can be rewritten as a QP in a form of

$$J^\star(x(t)) = \min_{U_{\text{ol}}} \tfrac{1}{2}U_{\text{ol}}^T H U_{\text{ol}} + x(t)^T F U_{\text{ol}} \tag{3.33a}$$

$$\text{s.t. } GU_{\text{ol}} \leq w + Ex(t). \tag{3.33b}$$

The term $x(t)^T Y x(t)$ is here omitted since it does not affect the optimal argument. The overall cost function at the optimum is then obtained by e.g. $\mathfrak{J}^\star(x(t)) = J^\star(x(t)) + x(t)^T Y x(t)$.

Remark 3.2.6 *MPC optimization problem defined as QP in* (3.33) *can be rewritten into its equivalent problem*

$$J^\star(x(t)) = \min_{\mathfrak{U}} \mathfrak{U}^T H \mathfrak{U} \tag{3.34a}$$

$$s.t.\ G\mathfrak{U} \leq w + Sx(t), \tag{3.34b}$$

by using substitution $\mathfrak{U} = U_{ol} + H^{-1}F^T x(t)$, *with* $S = E + GH^{-1}F^T$.

3.3 Stability

Even thought MPC has a great successes from its beginnings, researchers found out that there are many aspects, e.g. plant-model mismatch, disturbances, finite prediction horizon, which might drag the controlled system to instability and thus to hazardous situations. Therefore, for safety reasons, it was invariable to impose MPC optimization problem in a such manner which eliminates this shortcoming.

In the sequel, firstly stability and asymptotic stability (in Lyapunov sense) are defined followed by its application to MPC strategy.

3.3.1 Lyapunov Stability

In practice a standard convergence started to be only wanted, but not satisfactory property. It was desired not only to asymptotically converge e.g. to the origin, but also to persistently stay there (or its neighborhood) even after a perturbation. And this property is referred to as Lyapunov stability.

Given is a discrete-time autonomous system

$$x(t+1) = f(x(t)), \tag{3.35}$$

with $x(t) \in \mathbb{R}^n$, $f : \mathbb{R}^n \mapsto \mathbb{R}^n$ and $f(\mathbf{0}) = \mathbf{0}$.

Definition 3.3.1 *The origin of the system* (3.35) *is referred to be stable (in the sense of Lyapunov) if* $\forall \epsilon > 0$, $\exists \delta > 0$, *such that*

$$\|x(0)\| \leq \delta \implies \|x(t)\| < \epsilon, \ \forall t \geq 0$$

and additionally if the origin is locally attractive, i.e.

$$\|x(0)\| \leq \delta \implies \lim_{t \to \infty} \|x(t)\| = \mathbf{0},$$

then it is called asymptotically stable.

Theorem 3.3.2 *System* (3.35) *is stable if there exists a function* $V : \mathbb{R}^n \mapsto \mathbb{R}$ *for which*

$$V(x(t)) > 0, \ \forall x(t) \neq \mathbf{0} \tag{3.36a}$$

$$V(\mathbf{0}) = 0, \tag{3.36b}$$

$$V(x(t+1)) - V(x(t)) \leq 0, \ \forall x(t) \tag{3.36c}$$

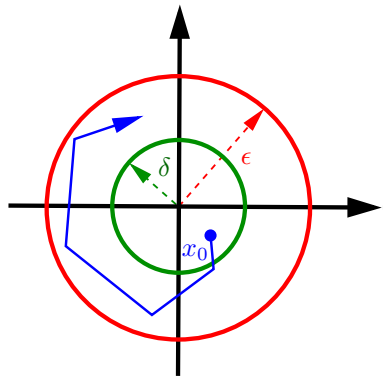


Figure 3.4: Illustrative example of a stable system.

holds. Moreover, if the function $V(x(t))$ *satisfies the strict inequality in* (3.36c), *i.e.* $V(x(t+1)) - V(x(t)) < 0.\ \forall x(t)$, *then the system* (3.35) *is said to be asymptotically stable.*

Definition 3.3.3 *The positive definite function* $V(\cdot)$ *which satisfies* (3.36) *is also referred as a Lyapunov function.*

3.3.2 Stability in MPC

In early years MPC policy did not explicitly ensured stability, yet it was achieved only by a proper tuning of engineers. This drawback motivated theoreticians to put a lot of attention to this topic. In late 1980's the necessity to use Lyapunov stability theory in MPC was proposed by (Keerthi and Gilbert, 1988) for discrete-time systems and by (Mayne and Michalska, 1990) for continous-time systems and from this point forward the cost function has been used as a natural Lyapunov function for ensuring closed-loop stability.

Consider general MPC optimization problem given as in (3.3), that is subjected to (3.2). The control objective is to stabilize and steer the system (3.3b) to the origin, i.e. $f(\cdot,\cdot) = \mathbf{0}$. Then the MPC optimization problem can be stated as

$$J^\star = \min_{U_{o1}} \ell_N(x_N) + \sum_{k=0}^{N-1} \ell(x_k, u_k) \tag{3.37a}$$

$$\text{s.t. } x_{k+1} = f(x_k, u_k),\ k = 0, \dots, N-1, \tag{3.37b}$$

$$x_k \in \mathcal{X},\ k = 0, \dots, N-1, \tag{3.37c}$$

$$u_k \in \mathcal{U},\ k = 0, \dots, N-1, \tag{3.37d}$$

$$x_N \in \mathcal{X}_f, \tag{3.37e}$$

$$x_0 = x(t), \tag{3.37f}$$

where x_k, u_k denote values of state and inputs predicted at the k-stage of the prediction horizon $N \in \mathbb{N}$, $x(t)$ is a state measurement, $\ell_N(x_N)$ is the terminal cost, $\ell(x_k, u_k)$ is the stage cost, $\mathcal{X}_f$ is terminal set and $\mathcal{X}$,$\mathcal{U}$ are polytopes, each of which contains the origin in its interior. Then MPC policy exhibits asymptotic closed-loop stability if following assumptions are met (Mayne et al., 2000):

A1: The terminal set satisfies the state constraints $\mathcal{X}_f \subset \mathcal{X}$, is closed and contains origin $\mathbf{0} \in \mathcal{X}_f$.

A2: Constraints are satisfied in $\mathcal{X}_f$ via a terminal feedback law, i.e. $\kappa_f(x_k) \in \mathcal{U}$, $\forall x_k \in \mathcal{X}_f$, $k = N, \dots, \infty$.

A3: The terminal set $\mathcal{X}_f$ is positive invariant under a terminal feedback $\kappa_f(x_k)$, i.e. $f(x_k, \kappa_f(x_k)) \in \mathcal{X}_f$, $\forall x_k \in \mathcal{X}_f$, $k = N, \dots, \infty$.

A4: Function $\ell_N(\cdot)$ is a local Lyapunov function, in other words $\ell_N(x_{k+1}) - \ell_N(x_k) \leq -\ell(x_k, \kappa_f(x_k))$, $\forall x_k \in \mathcal{X}_f$, $k = N, \dots, \infty$.

During the last decades a wide varieties of stabilizing MPC techniques have been proposed. Generally, their diversity stems from the modification of three essential ingredients that provide asymptotic stability (assumptions A1−A4), which are terminal penalty $\ell_N(\cdot)$, terminal constraint $\mathcal{X}_f$ and terminal feedback $\kappa_f(\cdot)$. In the sequel we briefly describe these methods. One can find an extensive overview in (Mayne et al., 2000) or in (de Oliveira Kothare and Morari, 2000; Keerthi and Gilbert, 1988; Rawlings and Muske, 1993; Scokaert and Rawlings, 1998).

Terminal equality constraint: One of the first methods used in MPC to ensure stability. Here $\ell_N((x_N) = 0$, $\mathcal{X}_f = \{\mathbf{0}\}$ is imposed to MPC. From A3 we

have that $\kappa_f(x_k) = \mathbf{0}$, $k = N, \ldots, \infty$. The main goal of this approach is to steer the controlled system to origin within N steps. This leads to decreased feasible set of MPC. Therefore it is required either to initialized the system in a neighborhood of the origin or to use a longer N.

Terminal constraint set: Here only terminal set $\mathcal{X}_f$ is embedded in MPC formulation and terminal cost $\ell_N((x_N) = 0$. The main goal is to employ two controllers. MPC which steers the system to the terminal set $\mathcal{X}_f$ within N steps, where the stabilizing terminal controller $\kappa_f(\cdot)$ takes over the control. Therefore this approach is also referred as a dual mode.

Terminal penalty: In this approach the terminal set is omitted $\mathcal{X}_f = \mathbb{R}^n$ and the stabilizing role is moved to terminal cost function $\ell_N(\cdot) \neq 0$. The downside of this methodology however is that it is valid only for smaller range of control problems (unless longer N is used).

Terminal penalty and terminal constraint set: One of the most used forms today, which exploits advantages from both stabilizing ingredients (terminal set $\mathcal{X}_f \subset \mathbb{R}^n$ with terminal penalty $\ell_N(\cdot) \neq 0$) and thus is suitable for wide range of control problems.

Infinite horizon: Infinite horizon ($N = \infty$) substitutes assumptions A1$-$A4, however leads to complex optimization problem (infinite number of optimization variables and constraints respectively).

Contractive constrains: Here the stabilizing essence is denoted be adding contractive constraints into MPC formulation, based on which the decaying values of states are explicitly enforced at each step, i.e.

$$\|f(x_k, \mu(x_k))\|_p \leq \gamma \|x_k\|_p,$$

with so-called concrative parameter $\gamma \in [0, 1)$ and $\|\cdot\|_p$ denoting p-norm.

Chapter 4

Explicit Model Predictive Control

As mentioned in Chapter 3, MPC is a widely adopted control strategy that has already found its usage in many industrial fields. The main advantage, which makes MPC industrially desirable, is a possibility to explicitly embed constraints into the control design problem, to handle multivariable systems with complex dynamics and last, but not least, to offer the best possible performance, which may correspond to the most profitable or efficient control. However, the drawback of this approach lies in computational complexity. This is due to the fact that its traditional implementation (RHC) relies on the use of a real-time optimization solver, which is required to compute the sequence of optimal control inputs for each new set of measurements within duration of one sample instant. Even though computational power and optimization algorithms are continuously improving (see references in Chapter 3), traditionally such solvers have been able only to deal with relatively low update rates. Hence, the conventional MPC applications have been limited to situations, where such software and hardware costs have been met and which allowed a sufficient time for solving the whole optimization problem. One could come across to this shortcoming of the conventional MPC control strategy for example in the automotive and aerospace industries, where systems (subject to actuator limitations) had to be sampled and controlled within range of milli- or micro-seconds.

Therefore, a different MPC implementation approach has been established at the beginning of this century. It is called an explicit MPC (Bemporad et al. (2002b))

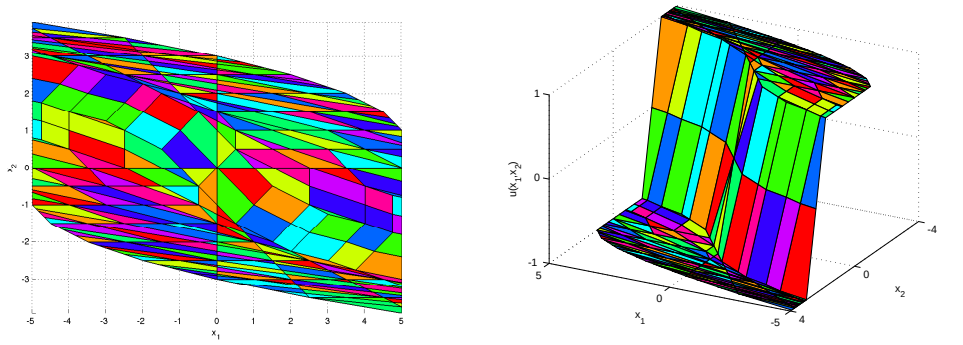
and it abolished the aforementioned required computational effort by shifting the whole optimization offline, where it could be computed by any computing device. Explicit MPC is based on a multiparametric programming (Borrelli (2003b); Gal and Nedoma (1972); Willner (1967)) that allows one to determine the optimal solution of an optimization problem for all feasible initial conditions as an explicit function of certain varying parameter(s). The repetitive optimization is here avoided, since the optimal solution can readily be obtained using the precomputed function, when the parameter(s) change. Thus the whole computational burden of obtaining optimal control inputs in the online procedure is reduced to a series of mere function evaluation. This eliminates the need of real-time optimization solvers and makes MPC accessible for applications with strictly limited computational resources such as in automotive (S.Di Cairano et al. (2012); Stewart and Borrelli (2008)) and aerospace (Di Cairano et al. (2012)) industries.

As numerous authors have shown (e.g. in Baotić et al. (2006); Bemporad et al. (2002a, 2003); Dua and Pistikopoulos (2000); Spjøtvold et al. (2005)), such a precomputed explicit solution, for a rich class of MPC problems, takes a form of a PWA function, which maps measurements of states onto optimal control actions. The domain of the PWA function is partitioned into a finite number of regions (also called critical regions), which are associated with affine functions that determine the optimal control input(s) for a given state measurement. A method that identifies the active region, hence an active control affine function, is a called the Point Location (PL) procedure, which consumes a major of time in the online implementation of explicit MPC. Figure 4.1 illustrates an example of an explicit solution for a 2-state system with one control input. Here, the planar domain (Figure 4.1(a)) consist of 278 critical regions, over which the associated pre-computed control law in the form of a PWA function is defined (Figure 4.1(b)).

The most prominent advantages of explicit MPC can be summarized as follows:

- Once the explicit controller is obtained, via computing the optimization problem offline, the online implementation requirements are decreased to a mere function evaluation (e.g. no matrix inversions are required). Needlessly to say, certification of those controllers is much cheaper (compared to the implicit approach).

- The low implementation cost of explicit controllers allows their usage even on simple hardware such as programmable logic controllers (PLC) or embedded



(a) Polytopic partition.

(b) Explicit representation of MPC feedback law.

Figure 4.1: Figure shows polytopic PWA explicit optimizer for a system with two states and one control input.

microchips.

- Reduced (online) computational effort enables explicit MPC feedbacks to be used on systems, where high update frequencies are required (e.g. aerospace and automotive industries)

- Explicit MPC controllers, e.g. example of which is shown in Figure 4.1, offer a better understanding of the control behavior and properties (comparing to the implicit approach). This is often exploited e.g. in safety analysis.

On the other hand, the explicit MPC implementation can be still prohibitively expensive for large optimization problems. Even though that the problem is solvable for a multitude of interesting control applications, the offline computation effort required to solve the multi-parametric optimization problem grows fast as the problem size increases. One should here point out that the complexity of such explicit optimizers grows hand in hand as the size of the optimization problem increases. As was shown by numerous researchers (e.g. Borrelli (2003b); Grieder (2004)), the complexity of explicit optimizers, expressed by the number of region over which the PWA control law is defined, can grow exponentially with the prediction horizon (i.e., with the number of optimized variables and constraints). Hence, one can conclude that the required storage space (as well as the online function evaluation time needed in PL problem) for the explicit MPC controller implementation

can easily overcome the hardware limitations, since the commonly used industrial low-cost hardware have their memory capacities strictly limited only up to a several kilobytes. Therefore, it is important to track this complexity of explicit optimizers in order to meet required limits.

4.1 Multiparametric Programming

As we have already mentioned, explicit model predictive control is based on multiparametric programming, where the optimal solution of an optimization problem is determined for a full range of parameter values (feasible initial conditions) in order to obtain an explicit representation of MPC feedback law. In this section we will discuss how these explicit solutions are constructed for both multiparametric linear programming (mp-LP) and multiparametric quadratic programming (mp-QP). Let us assume following optimization problem

$$J^\star = \min_z \; J(z, \theta) \tag{4.1a}$$

$$\text{s.t. } Gz \leq w + E\theta, \tag{4.1b}$$

with a vector of optimization variables $z \in \mathbb{R}^{n_z}$ and a vector of parameters $\theta \in \mathbb{R}^{n_\theta}$. The objective function $J(z, \theta)$ attains its optimum at $J^\star$. The constraints are formed by matrices $G \in \mathbb{R}^{r \times n_z}$, $w \in \mathbb{R}^r$ and $E \in \mathbb{R}^{r \times n_\theta}$, where r denotes the number of constraints.

Remark 4.1.1 *Note that in the previous Section 3.2.6 we have shown how to derive dense form of MPC formulated as a QP* (3.33)*, with decision variables* $U_{ol} \in \mathbb{R}^{Nm}$ *and parameter vector* $x(t) \in \mathbb{R}^n$*. Here, for generality, we denote optimization variables by* $z \in \mathbb{R}^{n_z}$ *as the control inputs might not be the only optimized variables (see e.g.* (3.22)*), i.e.* $U_{ol} \subseteq z$*. For the same reason the vector of all parameters is denoted by* $\theta \in \mathbb{R}^{n_\theta}$*, i.e.* $x(t) \subseteq \theta$*.*

In multiparametric programming we are interested to compute three components:

- The first component is a feedback law, which is also referred as an optimizer, i.e. $\mu^\star(\theta) = \arg\min_z \; J(z, \theta)$ w.r.t. $Gz \leq w + E\theta$, that takes a form of a polytopic PWA function.

- The second component is the objective function $J^\star(\theta)$ in a polytopic PWA form (if (4.1) is a mp-LP) or in a polytopic PWQ form (if (4.1) is a mp-QP).
- The third component is the polytopic partition, separated into polytopes (called regions), denoting a domain of all feasible parameters θ, i.e.

$$\Omega = \{\theta \in \mathcal{X} \mid \exists z : Gz \leq w + E\theta\}, \tag{4.2}$$

over which both functions $\mu^\star(\theta)$ and $J^\star(\theta)$ are being defined.

4.1.1 Categorization of Inequalities

Given is a set of r inequalities (4.1b). Let $\mathcal{I} = \{1, \ldots, r\}$ be the vector of indexes, i.e. $G_\mathcal{I} = G$ as it retains all of its rows (indexed by $\mathcal{I}$). Denote submatrices $G_\mathcal{A}$, $E_\mathcal{A}$, $w_\mathcal{A}$, $G_\mathcal{N}$, $E_\mathcal{N}$, $w_\mathcal{N}$ of matrices G, E and w, for $\mathcal{A} \subseteq \mathcal{I}$ and $\mathcal{N} \subseteq \mathcal{I}$ such that $\mathcal{A} \cup \mathcal{N} = \mathcal{I}$ and $\mathcal{A} \cap \mathcal{N} = \emptyset$. Moreover denote $n_\mathcal{A} = |\mathcal{A}|$ and $n_\mathcal{N} = |\mathcal{N}|$ as a number of active and inactive indicators, respectively.

Definition 4.1.2 (Active constraints) *An inequality is referred to be active if it holds with equality, for some $\theta \in \mathbb{R}^{n_\theta}$, at the optimum, i.e.*

$$\mathcal{A} = \{i \in \mathcal{I} \mid G_i z^\star - w_i - E_i \theta = 0\}. \tag{4.3}$$

Definition 4.1.3 (Inactive constraints) *An inequality is referred as an inactive if it holds with strict inequality, for some $\theta \in \mathbb{R}^{n_\theta}$, at the optimum,i.e.*

$$\mathcal{N} = \{i \in \mathcal{I} \mid G_i z^\star - w_i - E_i \theta < 0\}. \tag{4.4}$$

Definition 4.1.4 (Critical region) *A polytopic set of parameters θ associated to a particular feasible and optimal combination of indicators $\mathcal{A}$ is referred as a critical region and can be denoted by*

$$\mathcal{R} = \{\theta \in \Omega \mid A(\mathcal{A})\theta \leq b(\mathcal{A})\}. \tag{4.5}$$

Definition 4.1.5 (LICQ) *Let $\mathcal{A}$ be a set of active constraints, then we have that matrix $G_\mathcal{A}$ satisfies the* linear independence constraint qualification *(LICQ) if its gradients are linearly independent, i.e. matrix $G_\mathcal{A}$ is of full row rank.*

One should note here that violation of LICQ informs that in $G_\mathcal{A}$ are included redundant constraints, thus indicates a degeneracy.

4.1.2 Multiparametric Linear Programming

Assume an mp-LP program

$$J^\star(\theta) = \min_z \; c^T z \tag{4.6a}$$

$$\text{s.t. } Gz \leq w + E\theta, \tag{4.6b}$$

with decision variables $z \in \mathbb{R}^{n_z}$, parameters $\theta \in \mathbb{R}^{n_\theta}$ and vectors/matrices $c \in \mathbb{R}^{n_z}$, $G \in \mathbb{R}^{r \times n_z}$, $w \in \mathbb{R}^r$ and $E \in \mathbb{R}^{r \times n_\theta}$. In this section we will introduce the general concept how to synthesize a result of a mp-LP problem (4.6), which consist of:

- the optimizer $\mu^\star(\theta) = \arg\min_z J(z, \theta)$, w.r.t. $Gz \leq w + E\theta$
- the objective function $J^\star(\theta)$,
- and polytopic partition $\Omega = \cup_i \mathcal{R}_i$ consisting of M critical regions.

Since problem (4.6) is convex, the Karush-Kuhn-Tucker (KKT) conditions are necessary and sufficient conditions (Boyd and Vandenberghe (2004)) for solution $z^\star$ to be a global minimizer. Let us now state the Lagrangian function of the LP problem (4.6) as

$$\mathcal{L}(z, \theta, \lambda) = c^T z + \lambda^T (Gz - w - E\theta) \tag{4.7}$$

and the corresponding KKT conditions as

$$Gz^\star - w - E\theta \leq 0, \tag{4.8a}$$

$$\lambda^\star \geq 0, \tag{4.8b}$$

$$\lambda^{\star T}(Gz^\star - w - E\theta) = 0, \tag{4.8c}$$

$$c^T + \lambda^{\star T} G = 0, \tag{4.8d}$$

where $z^\star \in \mathbb{R}^{n_z}$ and $\lambda^\star \in \mathbb{R}^r$ are optimal solutions of primal and dual problem with zero duality gap. Further denote equation (4.8a) as the primal feasibility condition, equation (4.8b) as the dual feasibility condition, equation (4.8c) as the complementary slackness condition and equation (4.8d) as the stationary condition.

By separating KKT conditions (4.8) into active and inactive constraints, as in Section 4.1.1, one obtains

$$G_{\mathcal{A}} z^\star - w_{\mathcal{A}} - E_{\mathcal{A}}\theta = 0, \tag{4.9a}$$
$$G_{\mathcal{N}} z^\star - w_{\mathcal{N}} - E_{\mathcal{N}}\theta < 0, \tag{4.9b}$$
$$\lambda^\star_{\mathcal{N}} \geq 0, \tag{4.9c}$$
$$\lambda^\star_{\mathcal{A}} \geq 0, \tag{4.9d}$$
$${\lambda^\star_{\mathcal{A}}}^T (G_{\mathcal{A}} z^\star - w_{\mathcal{A}} - E_{\mathcal{A}}\theta) = 0, \tag{4.9e}$$
$${\lambda^\star_{\mathcal{I}}}^T (G_{\mathcal{N}} z^\star - w_{\mathcal{N}} - E_{\mathcal{N}}\theta) = 0, \tag{4.9f}$$
$$c^T + {\lambda^\star}^T G = 0. \tag{4.9g}$$

Here, by a simple investigation of (4.9c)−(4.9f) one can deduce that Lagrange multipliers associated with inactive constraints $\lambda_{\mathcal{I}}$ must be equal to zero, while in a case of active constraints Lagrange multipliers $\lambda_{\mathcal{A}}$ can be (but need not be) strictly positive, i.e.

$$\lambda^\star_{\mathcal{N}} = 0, \tag{4.10a}$$
$$\lambda^\star_{\mathcal{A}} > 0. \tag{4.10b}$$

To see this, notice that for all inactive constraints (4.4) we have that $G_{\mathcal{N}} z^\star - w_{\mathcal{N}} - E_{\mathcal{N}}\theta < 0$ and hence from (4.9f) we deduce that $\lambda^\star_{\mathcal{N}} = 0$.

By plugging (4.10) into (4.9) and omitting the complementary slackness in (4.9e) and(4.9f)[1]. Then the KKT conditions (4.9) simplifies into

$$G_{\mathcal{A}} z^\star - w_{\mathcal{A}} - E_{\mathcal{A}}\theta = 0, \tag{4.11a}$$
$$G_{\mathcal{N}} z^\star - w_{\mathcal{N}} - E_{\mathcal{N}}\theta < 0, \tag{4.11b}$$
$$\lambda^\star_{\mathcal{N}} = 0, \tag{4.11c}$$
$$\lambda^\star_{\mathcal{A}} > 0, \tag{4.11d}$$
$$c^T + {\lambda^\star_{\mathcal{A}}}^T G_{\mathcal{A}} = 0. \tag{4.11e}$$

By solving equation (4.11a) for $z^\star$ one can obtain optimizer

$$\mu^\star(\theta) = \alpha_z \theta + \beta_z, \tag{4.12}$$

[1]The complementary slackness condition in (4.9e) is trivially satisfied due to (4.9a) and (4.9f) is redundant due to (4.10a)

with

$$\alpha_z = G_{\mathcal{A}}^{-1} E_{\mathcal{A}},$$
$$\beta_z = G_{\mathcal{A}}^{-1} w_{\mathcal{A}}.$$

The dual optimizer can be derived from the stacionarity condition (4.11e) as

$$\lambda_{\mathcal{A}}^{\star} = -{G_{\mathcal{A}}^{-1}}^T c. \tag{4.14}$$

The critical region is then given as an intersection of all inequalities in (4.11). particularly it can be denoted by

$$\mathcal{R} = \mathcal{P}_{primal} \cap \mathcal{P}_{dual}, \tag{4.15}$$

hence as an intersection of $\mathcal{P}_{primal}$ defined by inactive primal condition (4.11b), where $z^\star$ is substituted by (4.12), and $\mathcal{P}_{dual}$ defined by active dual feasibility (4.11d). Subsequently we have that

$$(G_{\mathcal{N}}\alpha_z - E_{\mathcal{N}})\theta + G_{\mathcal{N}}\beta_z - w_{\mathcal{N}} < 0, \tag{4.16a}$$
$${G_{\mathcal{A}}^{-1}}^T c < 0. \tag{4.16b}$$

But, since equation (4.16b) does not contribute any restriction for parameter θ, it can be omitted and the critical region can be constructed by a closure of (4.16a), i.e. replacing $<$ by $\leq$, as

$$\mathcal{R} = \{x \in \mathbb{R}^n \mid Ax \leq b\}, \tag{4.17}$$

which is a polyhedron with matrices $A \in \mathbb{R}^{n_c \times n}$, $b \in \mathbb{R}^{n_c}$ defined by

$$A = \left[G_{\mathcal{N}}\alpha_z - E_{\mathcal{N}}\right], \; b = \left[w_{\mathcal{N}} - G_{\mathcal{N}}\beta_z\right],$$

where n_c denotes the number of (non-redundant) half-spaces.

To complete, the objective function (as a function of parameter θ) can be obtained via substitution of (4.12) into (4.6a), where we have that

$$J^\star(\theta) = \alpha_J \theta + \beta_J, \tag{4.18}$$

with matrices

$$\alpha_J = c^T \alpha_z,$$
$$\beta_J = c^T \beta_z.$$

Remark 4.1.6 *In the aforementioned procedure we are considering that matrix $G_{\mathcal{A}}$ is invertible. However this assumption does not always hold. If the matrix $G_{\mathcal{A}}$ is not invertible, then one can still apply QR decomposition, using of which will reflect in the computational complexity of the problem. Interested readers are referred to Borrelli et al. (2016).*

4.1.3 Multiparametric Quadratic Programming

In this section we will provide a general procedure how the mp-QP computes a given problem and yields a solution, which is composed by these three components:

- the optimizer $\mu^\star(\theta) = \arg\min_z J(z, \theta)$,
- the objective function $J^\star(\theta)$,
- and polytopic partition $\Omega = \cup_i \mathcal{R}_i$ consisting of $i = 1, \dots, M$ critical regions.

Consider a mp-QP program given as

$$J^\star(\theta) = \min_z \tfrac{1}{2} z^T H z + (F^T\theta + f_z)^T z + \theta^T Y \theta + f_\theta^T \theta + f_c \tag{4.20a}$$

$$\text{s.t. } Gz \leq w + E\theta, \tag{4.20b}$$

where $z \in \mathbb{R}^{n_z}$ is an optimized variable, $\theta \in \mathbb{R}^{n_\theta}$ is a parameter and $H \in \mathbb{R}^{n_z \times n_z}$, $H = H^T \succ 0$, $F \in \mathbb{R}^{n_\theta \times n_z}$, $f_z \in \mathbb{R}^{n_z}$, $Y \in \mathbb{R}^{n_\theta \times n_\theta}$, $f_\theta \in \mathbb{R}^{n_\theta}$, $f_c \in \mathbb{R}$, $G \in \mathbb{R}^{r \times n_z}$, $w \in \mathbb{R}^r$, $E \in \mathbb{R}^{r \times n_\theta}$. As it was shown in (Boyd and Vandenberghe, 2004), the KKT are necessary and sufficient conditions for solution $z^\star$ to be a global minimizer, since (4.6) is convex. Hence, we can state the Lagrangian function of the given problem (4.20) by

$$\mathcal{L}(z, \theta, \lambda) = \tfrac{1}{2} z^T H z + (F^T\theta + f_z)^T z + \theta^T Y \theta + f_\theta^T \theta + f_c + \lambda^T (Gz - w - E\theta) \tag{4.21}$$

and the KKT conditions can be formulated as

$$Gz^\star - w - E\theta \leq 0 \tag{4.22a}$$

$$\lambda^\star \geq 0 \tag{4.22b}$$

$$\lambda^{\star T}(Gz^\star - w - E\theta) = 0 \tag{4.22c}$$

$$Hz^\star + F^T\theta + f_z + G^T\lambda^\star = 0, \tag{4.22d}$$

where $z^\star \in \mathbb{R}^{n_z}$ and $\lambda^\star \in \mathbb{R}^r$ are optimal solutions of primal and dual problem with zero duality gap. Recalling from Section 4.1.2, the equation (4.22a) refers

the primal feasibility condition, equation (4.22b) the dual feasibility condition, equation (4.22c) the complementary slackness condition and equation (4.22d) the stacionarity condition. By introducing the conception of constrains categorization (see Section 4.1.1), where active constraints are denoted by index $\mathcal{A}$ and inactive by $\mathcal{N}$ as in (4.3) and (4.4), respectively, we can recast the KKT condition into form of

$$G_{\mathcal{A}} z^\star - w_{\mathcal{A}} - E_{\mathcal{A}}\theta = 0, \tag{4.23a}$$
$$G_{\mathcal{N}} z^\star - w_{\mathcal{N}} - E_{\mathcal{N}}\theta < 0, \tag{4.23b}$$
$$\lambda^\star_{\mathcal{A}} > 0, \tag{4.23c}$$
$$\lambda^\star_{\mathcal{N}} = 0, \tag{4.23d}$$
$$H z^\star + F^T\theta + f_z + G^T_{\mathcal{A}}\lambda^\star_{\mathcal{A}} = 0. \tag{4.23e}$$

Note that we have here, for the same reasons as in previous Section 4.1.2, omitted the complementary slackness (4.22c) condition.

To proceed, from the stationary condition (4.23e) we have that

$$\mu^\star(\theta, \lambda^\star) = -H^{-1}(F^T\theta + f_z + G^T_{\mathcal{A}}\lambda^\star_{\mathcal{A}}). \tag{4.24}$$

By substituting (4.24) into active primal feasibility (4.23a) one will obtain

$$-G_{\mathcal{A}}H^{-1}(F^T\theta + f_z + G^T_{\mathcal{A}}\lambda^\star_A) - w_{\mathcal{A}} - E_{\mathcal{A}}\theta = 0,$$

from which optimal Lagrange multipliers can be compactly expressed as

$$\lambda^\star_{\mathcal{A}} = \alpha_\lambda\theta + \beta_\lambda \tag{4.25}$$

with

$$\alpha_\lambda = (G_{\mathcal{A}}H^{-1}G^T_{\mathcal{A}})^{-1}(-G_{\mathcal{A}}H^{-1}F^T - E_{\mathcal{A}}),$$
$$\beta_\lambda = (G_{\mathcal{A}}H^{-1}G^T_{\mathcal{A}})^{-1}(-G_{\mathcal{A}}H^{-1}f_z - w_{\mathcal{A}}).$$

Finally the optimizer is obtained via substituting $\lambda^\star_A$ into (4.24) what yields

$$\mu^\star(\theta) = \alpha_z\theta + \beta_z \tag{4.27}$$

with

$$\alpha_z = H^{-1}G^T_{\mathcal{A}}\alpha_\lambda - H^{-1}F^T,$$
$$\beta_z = -H^{-1}f_z - H^{-1}G^T_{\mathcal{A}}\beta_\lambda.$$

Let us now construct a critical region which is given by (4.15), i.e. as an intersection of nactive primal conditions (4.23b) and active dual conditions (4.23c), where active Lagrange multipliers (4.25) and optimizer (4.27) are replaced. Subsequently a critical region, for a given vector of active constraints, can be defined as a closure of these inequalities, i.e. replacing $<$ by $\leq$, as

$$\mathcal{R} = \{\theta \in \mathbb{R}^{n_\theta} \mid A\theta \leq b\}, \tag{4.29}$$

which is a polyhedron with matrices $A \in \mathbb{R}^{n_\mathcal{H} \times n_\theta}$, $b \in \mathbb{R}^{n_\mathcal{H}}$ defined by

$$A = \begin{bmatrix} G_\mathcal{N}\alpha_z - E_\mathcal{N} \\ -\alpha_\lambda \end{bmatrix}, \; b = \begin{bmatrix} w_\mathcal{N} - G_\mathcal{N}\beta_z \\ \beta_\lambda \end{bmatrix},$$

where $n_\mathcal{H}$ denotes the number of (non-redundant) half-spaces.

The objective function $J(\theta)$ can be then easily obtained via substitution of the optimizer (4.27) into objective (4.20a). The substitution then yields

$$J^\star(\theta) = \theta^T \alpha_J \theta + \beta_J \theta + \gamma_J, \tag{4.30}$$

where

$$\begin{aligned} \alpha_J &= \tfrac{1}{2}\alpha_z^T H \lambda_z + F\alpha_z + Y, \\ \beta_J &= \beta_z^T H \alpha_z + \beta_z^T F^T + f_z^T \alpha_z + f_\theta, \\ \gamma_J &= \tfrac{1}{2}\beta_z^T H \beta_z + f_z^T \beta_z + f_c, \end{aligned}$$

what completes the solution of mp-QP problem.

Remark 4.1.7 *In the previous procedure we have not taken into account degeneration. In another words we have assumed that matrix $G_\mathcal{A}$ is of full row rank (LICQ) and that matrix $G_\mathcal{A}$ has at most m linearly independent rows.*

4.1.4 Properties of Multiparametric Solutions

In previous Sections 4.1.2 and 4.1.3 we have shown how to derive three essential ingredients of multiparametric solutions: the optimizer $\mu^\star(\theta)$, the objective function $J^\star(\theta)$, as functions of a parameter θ, and the underlying polytopic partition $\cup_i \mathcal{R}_i$. In the sequel we will summarize their properties.

Polytopic partition

The feasible set $\Omega \subset \mathbb{R}^{n_\theta}$ is partitioned into M critical regions $\mathcal{R}_i$ with

$\cup_i \mathcal{R}_i = \Omega$. The regions have disjoint interiors $\text{int}(\mathcal{R}_i) \cap \text{int}(\mathcal{R}_j) = \emptyset, \forall i \neq j$. The $i-$th critical region (for an appropriate set of parameters θ) is denoted by

$$\mathcal{R}_i = \{\theta \in \mathbb{R}^{n_\theta} \mid A_i\theta \leq b_i\}, \tag{4.32}$$

with matrices $A_i \in \mathbb{R}^{n_\mathcal{H} \times n_\theta}$ and $b_i \in \mathbb{R}^{n_\mathcal{H}}$ defined in (4.1.2) and (4.1.3), respectively. Furthermore, $n_\mathcal{H}$ is the number of half-spaces defining the i-th region.

Optimizer

The optimizer $\mu^\star : \Omega \to \mathbb{R}^{n_z}$ is a continuous PWA function defined over polytopic partition Ω, hence it is a polytopic PWA function in the sense of Def. 2.3.1, denoted as

$$\mu^\star(\theta) = \begin{cases} F_{z,1}\theta + g_{z,1} & \text{if } \theta \in \mathcal{R}_1 \\ \vdots & \\ F_{z,M}\theta + g_{z,M} & \text{if } \theta \in \mathcal{R}_M, \end{cases} \tag{4.33}$$

with matrices $F_z \in \mathbb{R}^{n_z \times n_\theta}$ and $g_z \in \mathbb{R}^{n_z}$.

Objective function

In mp-LP the objective function $J^\star : \Omega \to \mathbb{R}$ is a continuous and convex PWA function defined over polytopic partition Ω, thus a polytopic PWA function (cf. Def 2.3.1), denoted by

$$J^\star(\theta) = \begin{cases} F_{J,1}\theta + g_{J,1} & \text{if } \theta \in \mathcal{R}_1 \\ \vdots & \\ F_{J,M}\theta + g_{J,M} & \text{if } \theta \in \mathcal{R}_M, \end{cases} \tag{4.34}$$

with a matrix $F_J \in \mathbb{R}^{n_\theta}$ and a scalar $g_J \in \mathbb{R}$.

In mp-QP the objective function $J^\star : \Omega \to \mathbb{R}$ is a continuous and convex PWQ function defined over polytopic partition Ω, i.e. polytopic PWQ function (see Def. 2.3.2), denoted as

$$J^\star(\theta) = \begin{cases} \theta^T H_{J,1}\theta + F_{J,1}\theta + g_{J,1} & \text{if } \theta \in \mathcal{R}_1 \\ \vdots & \\ \theta^T H_{J,M}\theta + F_{J,M}\theta + g_{J,M} & \text{if } \theta \in \mathcal{R}_M, \end{cases} \tag{4.35}$$

with a matrix $H_J \in \mathbb{R}^{n_\theta \times n_\theta}$, $F_J \in \mathbb{R}^{n_\theta}$ and a scalar $g_J \in \mathbb{R}$.

Theorem 4.1.8 *The optimizers $\mu^\star(\theta)$ defined in* (4.12) *or in* (4.27) *are continuous PWA functions defined over a feasible set Ω. The value function $J^\star(\theta)$ given in a form of* (4.18) *is a continuous and convex PWA function defined over Ω, while in mp-QP the value function* (4.30) *is a continuous and convex PWQ function defined over Ω. The critical regions* (4.32) *and* (4.29)*, respectively, are polytopes with disjoint interiors, the union of which covers the entire set of feasible parameters $\cup_i \mathcal{R}_i = \Omega$, $\forall i = 1, \ldots, M$, where M denotes number of critical regions. The set Ω is hence polytopic partition defined as in Definition 2.2.12.*

Proof. The proof can be seen in Borrelli et al. (2016).

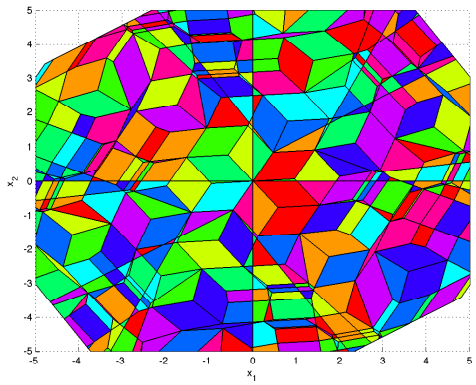
To illustrate results of the multiparametric programming we have devised MPC controller with model $x_{k+1} = \left[\begin{smallmatrix} 0.5403 & -0.8415 \\ 0.8415 & 0.5403 \end{smallmatrix}\right] x_k + \left[\begin{smallmatrix} -0.4597 \\ 0.8415 \end{smallmatrix}\right] u_k$, describing dynamics of a pendulum, constraints $-5 \leq x \leq 5$, prediction horizon $N = 5$, weighting matrices $Q_x = I_2$, $Q_u = 1$ and norms 1-norm, 2-norm to cover both mp-LP and mp-QP scenario. All multiparametric results (polytopic partition, optimizer and objective function) are depicted in in Figure 4.2. Here one should note the complexity difference (in terms of number of regions over which these functions are defined) between mp-LP (440 regions) and mp-QP (93 regions), which stems from the increased number of optimized variables and constraints (cf. Section 3.2.4).

4.2 Multiparametric Algorithms

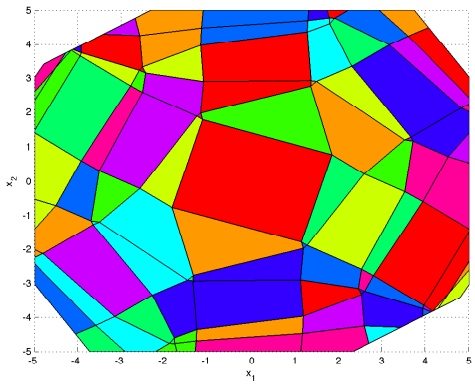
The goal of all multiparametric algorithms is to synthesize the polytopic partition Ω, which is separated into M critical regions $\mathcal{R}_i$, $i = 1, \ldots, M$, and both objective function $J^\star(\theta)$ and optimizer $\mu^\star(\theta)$ that are defined over the aforementioned polytopic partition. In other words, they need to determine all components defined in Section 4.1.4. All multiparametric algorithms are composed of two parts:

Active set generator: which aims at determining all active (4.3) constraints that would generate all full-dimensional critical regions (4.5).

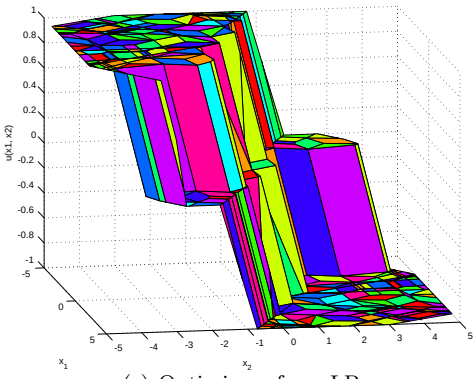
KKT solver: that constructs all critical regions, via using the list of active constraints formed in the active set generator, that would cover the entire feasible set of parameters Ω. The principle of this solver is described in Section 4.1.2 and 4.1.3, respectively.



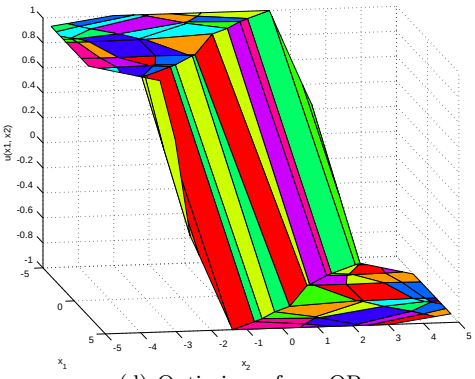
(a) Polytopic partition of mp-LP.



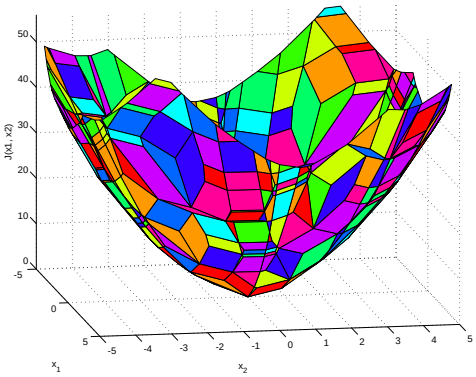
(b) Polytopic partition of mp-QP.



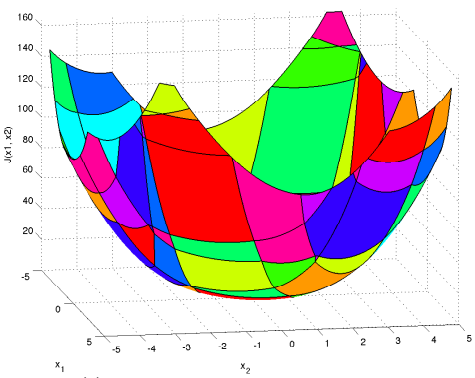
(c) Optimizer of mp-LP.



(d) Optimizer of mp-QP.



(e) Objective function of mp-LP.



(f) Objective function of mp-QP.

Figure 4.2: Illustrative results of mp-LP and mp-QP.

Since the KKT solver is practically the same for all algorithms, hence we will concentrate at the active set generators. In sequel we will introduce three commonly used multiparametric algorithms. Particularly, we will introduce the enumeration based method (Gupta et al., 2011), the geometric method (Baotić, 2002), embeded in MPT toolbox (Herceg et al., 2013a), and a geometric approach (Bemporad et al., 2002b)

4.2.1 Enumeration Based Approach

The active set generator in the explicit enumeration based approach is straightforwardly constructed by a simple listing of all possible combinations of active sets. Actually, this was the first approach in multiparametric programming, which was however criticized by its limited applicability due to its computational complexity. The main drawback was that only a fragment of the entire combination tree yielded full-dimensional critical regions inside of Ω out of the explored space $\mathcal{X}$, thus a lot of unnecessary computation was required. Moreover the combination tree could easily become too complex even for relatively small problems. Therefore, for a long time, the research community was not paying any attention to this technique.

Recently, however, this enumeration based technique was "reinvented" by Gupta et al. (2011). Its authors have shown that this approach can be enhanced by a pruning criteria, which significantly reduces the total number of active sets (needed to be enumerated), while also providing a guarantee that its solution will arrive at the minimal number of partitions (critical regions) while ensuring that the whole Ω will be explored. In C. Feller (2013) the authors have analyzed this technique and shown its performance, compared to the commonly used geometric approach (Baotić, 2002), on two illustrative examples. It was shown that this enumeration based technique is very practical, for MPC problems with simple constraints and prediction horizon, even when the state space is large (e.g. with 80 states). Note that this approach has been further carefully studied and many researchers are trying to further enhance this technique (see e.g. P. Ahmadi-Moshkenani and Johansen (2016), where facet-to-facet property has been exploited).

The principle of this approach will be explained on two examples. In the first one we illustrate the applicability of this approach in LP is pure, while in the second example we fully demonstrate its potential.

Consider two dimensional mp-LP as in (4.6), where $z \in \mathbb{R}^{n_z}$ with $s = 2$

(e.g. single input $m = 1$ and prediciton horizon $N = 2$) and five constraints $r = 5$. From the fact that we are dealing with LP problem we can conclude that the optimum will be located at the vertex of (4.6b), hence in the intersection of two constraints (see LP properties in Section 2.2.3). Therefore, we need to generate list of active sets only for all couples of active constraints. (In general we need to produce n_z-tumples of all available constraints.) Thus, our list of all possible active sets as in (4.3) will be defined by

$$\mathcal{W} = \{\mathcal{A}_1 = \{1, 2\}, \ldots, \mathcal{A}_4 = \{1, 5\}, \mathcal{A}_5 = \{2, 3\}, \ldots, \mathcal{A}_{10} = \{4, 5\}\}. \tag{4.36}$$

Now comes the crucial part, where we need to determine if combinations $\mathcal{A}_i, \forall i = 1, \ldots, 10$ in $\mathcal{W}$ are optimal. Authors in Gupta et al. (2011) proposed to use LP in a form

$$\begin{aligned}
\max_{t,z,\theta,\lambda^\star_{\mathcal{A}_i}} \quad & t && \text{(4.37a)}\\
\text{s.t. } & c^T + {\lambda^\star_{\mathcal{A}_i}}^T G_{\mathcal{A}_i} = 0, && \text{(4.37b)}\\
& G_{\mathcal{A}_i} z - w_{\mathcal{A}_i} - E_{\mathcal{A}_i}\theta = 0, && \text{(4.37c)}\\
& t \leq w_{\mathcal{N}_i} + S_{\mathcal{N}_i}\theta - G_{\mathcal{N}_i} z, && \text{(4.37d)}\\
& \lambda^\star_{\mathcal{A}_i} \geq t, && \text{(4.37e)}\\
& t \geq 0, && \text{(4.37f)}
\end{aligned}$$

with decision variables $t, z, \theta, \lambda^\star_{\mathcal{A}_i}$, $\mathcal{N}_i = \mathcal{I}\backslash\mathcal{A}_i$. We have that (4.37) is feasible if $t^\star > 0$ what indicates that the active set $\mathcal{A}_i$ is feasible optimal and yields a full-dimensional critical region. Thus $\mathcal{A}_i$ is preserved in the list $\mathcal{W}$. On the other hand, if (4.37) is infeasible, then $\mathcal{A}_i$ is not optimal and can be directly removed from the list $\mathcal{W}$. We would like to note, that since we are dealing here with mp-LP problem, there is no need to check whether $\mathcal{A}_i$ is at least feasible (and hence to employ the pruning technique from Gupta et al. (2011)). By iterative computation of all ten combinations, the entire list $\mathcal{W}$ will be explored and only optimal combinations of active constraints will be stored. Hence part of the active set generator is finished and the priority is then handovered to KKT solver, where mp-LP algorithm will synthesize polytopic partition, objective function and optimizer as in Section 4.1.4 via method described in Section 4.1.2.

Remark 4.2.1 *Advantage of this mp-LP solver based on the explicit enumeration technique is that we need to explore only n_z-tumples of all constraints. Therefore,*

this method could be used for processes with e.g. single input ($m = 1$) and twenty states $n_\theta = n = 20$. But the limitation is more strict because if e.g. 1-norm is considered (see Section 3.2.3) in MPC problem as in (3.22)*, then the number of optimized variables will be here automatically increased from $n_z = Nm$ to $\hat{n}_z = (n_z + Nm + Nn)$ and number of constraints from r to $\hat{r} = (r + 2Nm + 2Nn)$, where N denotes prediction horizon, m number of inputs and n number of states* [2]*. By considering the aforementioned setup, we would end up with $\hat{n}_z = 24$ optimization variables and $\hat{r} = 89$ constraints, which would result in computational explosion of such mp-LP algorithm.*

The contribution of Gupta et al. (2011) is in mp-LP algorithms omitted, hence we will provide an additional example. Consider mp-QP problem given by (4.20), where we denote n_z dimension of optimized variables, r as number of constraints and n_θ as dimension of parameter θ. From (2.2.4) we have that the optimum of QP may be located anywhere, at the face or in the interior, of Ω. Therefore, the list of all active sets will be defined as

$$\begin{aligned}\mathcal{W} = \{\mathcal{A}_1 = \{\}, \mathcal{A}_2 = \{1\}, \ldots, \mathcal{A}_{r+1} = \{r\}, \ldots,\\ \mathcal{A}_{r+2} = \{1,2\}, \ldots, \mathcal{A}_{n_A} = \{1,2,\ldots,r\}\},\end{aligned}$$

where $n_A = \binom{r}{n_z}$, hence the maximal number of optimal active sets can be $k = \sum_{i=0}^{n_z} \binom{r}{i}$. Therefore, in our case we have to explore all k candidates of active sets via LP problem

$$\max_{t,z,\theta,\lambda^\star_{\mathcal{A}_i}} \quad t \tag{4.39a}$$

$$\text{s.t. } H z^\star + F\theta + f + {G_{\mathcal{A}_i}}^T \lambda^\star_{\mathcal{A}_i} = 0, \tag{4.39b}$$

$$G_{\mathcal{A}_i} z - w_{\mathcal{A}_i} - E_{\mathcal{A}_i}\theta = 0, \tag{4.39c}$$

$$t \le w_{\mathcal{N}_i} + S_{\mathcal{N}_i}\theta - G_{\mathcal{N}_i} z, \tag{4.39d}$$

$$\lambda^\star_{\mathcal{A}_i} \ge t, \tag{4.39e}$$

$$t \ge 0, \tag{4.39f}$$

[2] If 1−norm is considered in MPC then we need to add additional optimization (slack) variables ϵ^x and ϵ^u. Particularly Nn variables for term $\|Q_x x_k\|_1$ and Nm variables for term $\|Q_u u_k\|_1$. The constraints are also extended by the aforementioned slacks variables, e.g. by $2n$ for each predicted step $\pm\epsilon^x_k \le Q_x x_k$ and by $2m$ for each $\pm\epsilon^u_k \le Q_u u_k$, what is together $(2Nn + 2Nm)$ constraints. Also note that the terminal penalty (e.g. $\|Px_N\|_1$) is here omitted.

with decision variables $t, z, \theta, \lambda^{\star}_{\mathcal{A}_i}$, $\mathcal{N}_i = \mathcal{I} \backslash \mathcal{A}_i$. This modified problem then indicates whether the given active set $\mathcal{A}_i$ is feasible optimal ($t^{\star} > 0$). If we will compare mp-QP problem, with the same setup $n_z = 2$ and $r = 5$, where $k = 16$ with mp-LP problem where $k = 10$, then one can say that we are disadvantaged, in a sense that we need to evaluate more combinations of active constraints. However, the main added value of Gupta et al. (2011) lies in the pruning technique, which can tackle this problem that has arisen.

The concept of the proposed pruning technique relies on the fact that examined set of active constrains can be either feasible optimal, feasible or infeasible. Particularly, if active set $\mathcal{A}_i$ is not feasible optimal, i.e. LP problem (4.39) is infeasible, then the same LP problem can be solved once again but with omitted stationarity condition (4.39b). Then if this modified problem is also infeasible, then we have that $\mathcal{A}_i$ and all its subsets are infeasible and can be removed from the list $\mathcal{W}$.This technique can significantly decrease the total number of active set, which the algorithm need to explore, thus to reduce the computational complexity of this enumeration based approach.

And the whole iterative procedure is summarized in the Algorithm 1 and the pruning technique is illustrated in Figure 4.3. Here the algorithm determined that active set combination $\mathcal{A}_i = 3$ is infeasible, hence all subsequent subsets $\mathcal{A}_i = \{\star 3 \star\}$ were removed from the list $\mathcal{W}$.

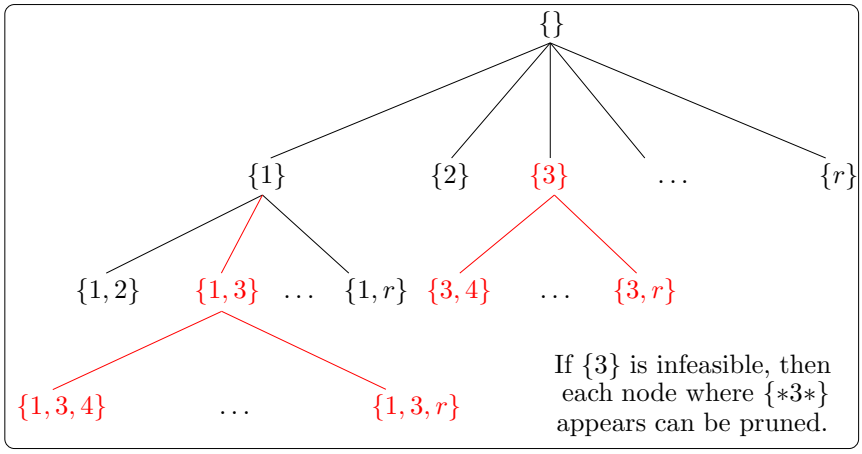


Figure 4.3: Pruning in enumeration strategy used in Gupta et al. (2011).

Algorithm 1: Enumeration based approach.

Input: List of all candidates: $\mathcal{W} = \mathcal{A}_i$, $i = 1, \ldots, \sum_{i=0}^{n_z} \binom{r}{i}$

Output: List of all optimal candidates: $\mathcal{W}^\star = \mathcal{A}_j$

Initialization: $i \leftarrow 1$, $k \leftarrow \sum_{i=0}^{s} \binom{r}{i}$;
while $i \leq k$ **do**
  Extract i-th element from the list $\mathcal{W}$: $\mathcal{A}_i \leftarrow \mathcal{W}_i$;
  if $G_{\mathcal{A}_i}$ *has full row rank* **then**
    Solve LP problem (4.39);
    if *feasible* **then**
      $i \leftarrow i + 1$;
    else
      Solve LP problem (4.39) with dropped (4.39b);
      if *feasible* **then**
        Remove combination $\mathcal{A}_i$: $\mathcal{W} \leftarrow \mathcal{W} \backslash \mathcal{A}_i$;
        $k \leftarrow k - 1$;
      else
        Pruning: remove all combinations of $\mathcal{A}_i$: $\mathcal{W} \leftarrow \mathcal{W} \backslash \{\star \mathcal{A}_i \star\}$;
        $k \leftarrow |\mathcal{W}|$;
      end
    end
  else
    Remove combination $\mathcal{A}_i$: $\mathcal{W} \leftarrow \mathcal{W} \backslash \mathcal{A}_i$;
    $k \leftarrow k - 1$;
  end
end
return $\mathcal{W}^\star \leftarrow \mathcal{W}$;

One should note that the shown Algorithm 1 is simplified. This is due to the fact that removing sequence $\mathcal{A}_i$ from the entire list of $\mathcal{W}$ can be costly operation, since it can be performed multiple times. Therefore, it is more efficient to save the infeasible combinations and enforce the algorithm to construct iteratively $\mathcal{W}$ without the mentioned infeasible combinations. Subsequently, as the enumeration algorithm is finished, set of all optimal active sets $\mathcal{W}^\star$ is determined, KKT solvers then takes priority and computes all results shown in Section 4.1.4.

Remark 4.2.2 *The pruning technique can be even more enhanced via taking into account all inequalities that can not be satisfied at the same time. For example consider these two constraints*

$$-a \leq z \leq a \tag{4.40}$$

that we can equivalently formulate as

$$z \leq a, \tag{4.41a}$$

$$-z \leq a. \tag{4.41b}$$

It is obvious that if constraint (4.41a) *is active, then constraint* (4.41b) *can not be active (and vice versa). Therefore, we can conclude that their combination is incompatible, hence this combination (in any level) can be immediately removed from list* $\mathcal{W}$.

4.2.2 Multiparametric Algorithm I

Multiparametric algorithm of Baotić (2002), which can be categorized as a geometric approach, tries to avoid the combination explosion of the complete enumeration via applying recursive exploration strategy in order to explore only feasible set of parameters Ω and not the whole set $\mathcal{X}$. Therefore, evaluation of a lot of infeasible active set candidates is here avoided, what was in Gupta et al. (2011) handled by pruning technique.

Given is problem (4.1), with parameters subjected to $\theta \in \mathcal{X}$. Denote an arbitrary feasible parameter $\theta_0 \in \mathcal{X}$, which is commonly chosen as a center of $\mathcal{X}$, $\mathcal{Q} = \theta_0$ to be a set of all unexplored candidates and $\mathcal{Q}_i$ as the i-th candidate of set $\mathcal{Q}$. Algorithm of (Baotić, 2002) proceeds as follows. Extract a candidate θ_0 from $\mathcal{Q}$, i.e. $\theta_0 = \mathcal{Q}_1$. For the given parameter θ_0 solve the multiparametric optimization problem (4.1). If this problem is feasible, then the optimizer $\mu^\star(\theta_0)$ is obtained. Subsequently, by plugging $\mu^\star(\theta_0)$ into primal feasibility condition of (4.1), i.e. (4.8a) or (4.23a), we determine set of all active constraints $\mathcal{A}_{\theta_0}$. With the set of active constraints in a hand one can compute the optimizer $\mu_i^\star(\theta)$, the objective function $J_i^\star(\theta)$ and the critical region $\mathcal{R}_i = \{\theta | A_j\theta \leq b_j\}$, where $j = 1, \ldots, n_\mathcal{F}$ denotes the j-th half-space of $\mathcal{R}_i$, via procedure shown in Section 4.1.2 and Section 4.1.3, respectively. For each face $\mathcal{F}_j = \{\theta | A_j\theta = b_j\}$ of the critical region $\mathcal{R}_i$ compute the center $\bar{\theta}_j \in \mathcal{F}_j$ and shift it by a scalar $\epsilon > 0$ in direction of the normal vector A_j, i.e. $\theta_j = \bar{\theta}_j + \epsilon(A_j)^T$. Next, remove all redundant candidates from the

vector θ_j that are contained in already created regions, i.e. $\theta_j = \{\}$ if $\theta_j \in \cup_i \mathcal{R}_i$, $\forall j$. Finally, discard examined parameter $\mathcal{Q} = \mathcal{Q} \backslash \theta_0$ and store all unexplored candidates $\mathcal{Q} = \mathcal{Q} \cup_j \theta_j$. The algorithm then proceed iteratively by extracting a next candidate parameter $\theta_0 = \mathcal{Q}_1$, until the set of all candidates is empty $\mathcal{Q} \neq \{\}$. The result of this algorithm is objective function $J^\star(\theta)$ and optimizer $\mu^\star(\theta)$ defined over polytopic partition Ω, according to Section 4.1.4. The procedure of this approach Baotić (2002) is summarized in Algorithm 2 and illustrated in Figure 4.4.

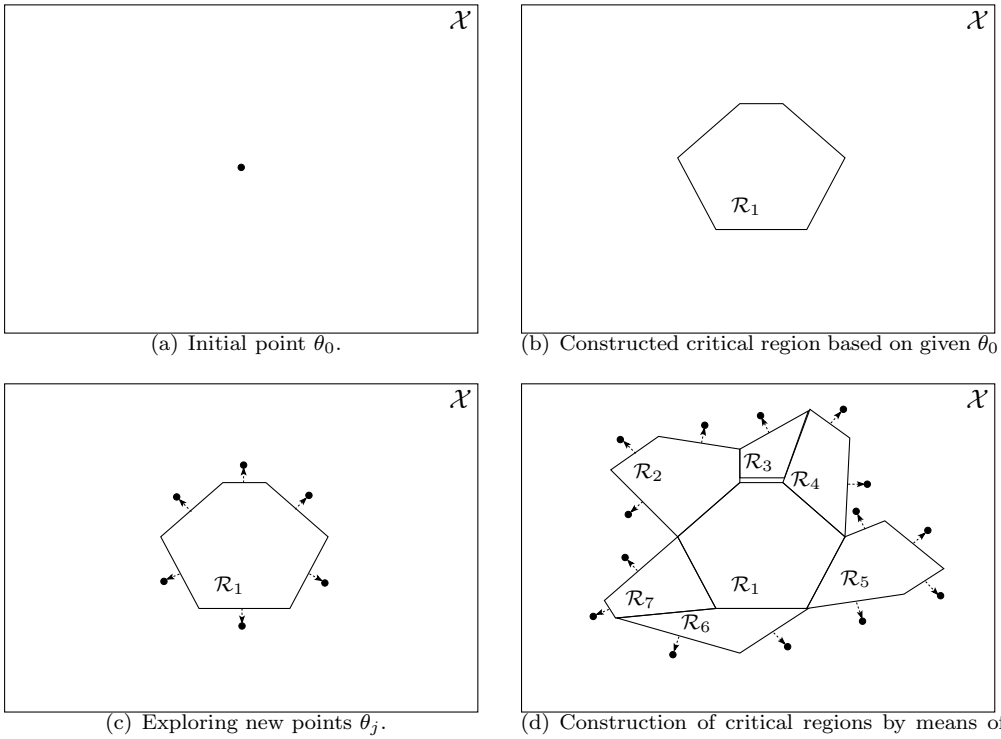
Algorithm 2: Algorithm of Baotić (2002).

Input: $\mathcal{X}$

Output: Ω, $J^\star(\theta)$, $\mu^\star(\theta)$

Initialization: $\mathcal{Q} \leftarrow \theta_0$, $i \leftarrow 0$;
while $\mathcal{Q} \neq \{\}$ **do**
  Extract the first vector of parameters: $\theta_0 \leftarrow \mathcal{Q}_1$;
  Solve (4.1) with $\theta = \theta_0$ and obtain $\mu^\star(\theta_0)$;
  if *infeasible* ***or*** $\theta_0 \in \cup_i \mathcal{R}_i$ **then**
    $\mathcal{Q} \leftarrow \mathcal{Q} \backslash \theta_0$;
  else
    Update increment: $i \leftarrow i + 1$;
    Find active sets:$\mathcal{A}_{\theta_0} \leftarrow \{k \in \mathcal{I} \mid G_k \mu^\star(\theta_0) - w_k - E_k \theta_0 = 0\}$;
    Synthesize: $\mu_i^\star(\theta), J_i^\star(\theta)$ and $\mathcal{R}_i = \{\theta | A_j \theta \leq b_j\}$, $j = 1, \ldots, n_\mathcal{F}$;
    for $j = 1, \ldots, n_\mathcal{F}$ **do**
      Find a center of the face: $\bar{\theta}_j \in \mathcal{F}_j = \{\theta | A_j \theta = b_j\}$;
      Create a new point: $\theta_j \leftarrow \bar{\theta}_j + \epsilon (A_j)^T$, with $\epsilon > 0$;
      Remove redundant candidates: $\theta_j \leftarrow \{\}$ if $\theta_j \in \cup_i \mathcal{R}_i$;
    end
    Remove the examined candidate: $\mathcal{Q} \leftarrow \mathcal{Q} \backslash \theta_0$;
    Store all new candidates: $\mathcal{Q} \leftarrow \mathcal{Q} \cup_j \theta_j$;
  end
end
return $\Omega \leftarrow \cup_i \mathcal{R}_i, \mu^\star(\theta) \leftarrow \cup_i \mu_i^\star(\theta), J^\star(\theta) \leftarrow \cup_i J_i^\star(\theta)$;

The main advantage of this approach provided by Baotić (2002) is that only the regions from the interested feasible set Ω are being investigated, hence the computational complexity of such algorithm is bearable. A comparison of this approach and the aforementioned enumeration technique from Gupta et al. (2011)



(a) Initial point θ_0.

(b) Constructed critical region based on given θ_0.

(c) Exploring new points θ_j.

(d) Construction of critical regions by means of given θ_j.

Figure 4.4: Figure illustrates the Algorithm 2. Here the black dots represent the points θ_j stored in the set $\mathcal{Q}$, based on which critical regions $\mathcal{R}$ are constructed.

is shown e.q. in C. Feller (2013). On the other hand, the disadvantage of this technique is that there is no theoretical guarantee that the union of all created critical regions will cover the entire feasible set of all parameters $\Omega = \cup_i \mathcal{R}_i$. This is due to ϵ, that is used to obtain new candidates θ_j (by shifting central points of faces $\bar{\theta}_j \in \mathcal{F}_j$). Even though that this scalar is typically set as a very small number (e.g. 10^{-5}), there is possibility that in this tiny area actually exist another critical region(s), thus we have that $\Omega \supseteq \cup_i \mathcal{R}_i$ (see e.g. Figure 4.4(d), where such gap is created under the region $\mathcal{R}_3$). Moreover, since only center of a face is considered, this might also lead to not exploring certain areas, hence to create gaps in Ω.

Remark 4.2.3 *In Algorithm 2 one can come across to critical regions, which are*

not full-dimensional (degenerated), what can mean e.g. that the point x_0 lies in the facet of another critical region. Then an additional action is needed to be performed, e.g. perturbation that further shifts the point θ_0, in any direction, until the critical region is fully-dimensional.

4.2.3 Multiparametric Algorithm II

In previous section we have shown algorithm which uses geometry in order to construct the explicit solution. Specifically, critical regions were derived based on exploring faces of already constructed regions. In what follows, we present algorithm proposed in Bemporad et al. (2002b), the main idea is also to exploit geometry to synthesis critical regions, but with a different exploration of feasible space $\mathcal{X}$.

Consider that we are given problem (4.1) with parameters subjected by $\theta \in \mathcal{X}$. Denote $\mathcal{Q} = \{\mathcal{Q}_1, \dots \mathcal{Q}_M\}$ to be the list of all critical region candidates. Algorithm in (Baotić, 2002) then proceeds as follows. Initialization of algorithm is accomplished by assigning to the set of all unexplored candidates $\mathcal{Q}$ entire feasible domain, i.e. $\mathcal{Q} = \mathcal{X}$. Then an investigated region $\mathcal{Q}_0 = \{\theta \mid H^i_{\mathcal{Q}}\theta \leq K^i_{\mathcal{Q}}\}$, with half-spaces $i = 1, \dots, n_{H_{\mathcal{Q}}}$, is extracted from the set $\mathcal{Q}$. Generally, it does not matter which one is chosen, but usually the first one $\mathcal{Q}_1$ is chosen, i.e. $\mathcal{Q}_0 = \mathcal{Q}_1 = \mathcal{X}$. Next, for this particular candidate $\mathcal{Q}_0$, a feasible parameter θ_0 is determined such that $\theta_0 \in \mathcal{Q}_0$. This can be done by solving one LP

$$\max_{\theta,z,\epsilon} \ \epsilon \tag{4.42a}$$

$$\text{s.t. } H^i_{\mathcal{Q}}\theta + \epsilon\|H^i_{\mathcal{Q}}\|_2 \leq K^i_{\mathcal{Q}}, \ i = 1, \dots, n_{H_{\mathcal{Q}}} \tag{4.42b}$$

$$Gz - E\theta - w \leq 0. \tag{4.42c}$$

Problem in (4.42) is basically enhanced Chebychev problem (2.20), by primal feasibility constraints (4.42c). This particular constraints are here embedded in order to restrict the parameter-space only to Ω, hence the redundant regions are not explored. By recalling from Section 2.2.3, the problem (4.42) can return three different results from which we can deduce that:

- If the optimized radius of the ball $\epsilon > 0$, then the investigated critical region is full-dimensional.

- If the optimized radius of the ball $\epsilon = 0$, then the investigated critical region is lower-dimensional.
- If the optimized radius of the ball $\epsilon < 0$, then the investigated critical region is empty.

Assume, that the solution of the problem (4.42) returns the center θ_0 of the full-dimensional critical region $\theta_0 \in \mathcal{Q}_0$. In order to obtain the optimizer $\mu^\star(\theta_0)$ we solve (4.1) for the given parameter θ_0. With θ_0 in hand, we solve the problem (4.1) what leads to the optimizer $\mu^\star(\theta_0)$. The vector of all active constraints $\mathcal{A}_{\theta_0}$ is obtained by solving KKT primal feasibility condition, with θ_0 and the optimizer $\mu^\star(\theta_0)$. Subsequently, an objective function $J^\star(\theta)$, an optimizer $\mu^\star(\theta)$ and a critical region $\mathcal{R} \subseteq \mathcal{Q}_0$ are computed for the neighborhood of the parameter θ_0 via procedure shown in Section 4.1.2 and 4.1.3, respectively. Removing the created critical region $\mathcal{R}$ from the investigated set $\mathcal{Q}_0$ typically leads to an unconvex set. Yet, one can still use a set difference technique $\mathcal{Q}_0 \backslash \mathcal{R}$ (cf. Definition 2.2.13), which will tessellate the $\mathcal{Q}_0$ into collection of k candidates, i.e. $\mathcal{Q} \backslash \mathcal{Q}_0 = \{\mathcal{Q}_{n_\mathcal{Q}}\}_{n_\mathcal{Q}=1}^{k}$. Subsequently, the examined region $\mathcal{Q}_0$ is removed $\mathcal{Q} = \mathcal{Q} \backslash \mathcal{Q}_0$ and all new candidates are added into the list $\mathcal{Q} = \mathcal{Q} \cup_{n_\mathcal{Q}} \mathcal{Q}_{n_\mathcal{Q}}$. Algorithm then proceeds iteratively by selecting a next candidate from $\mathcal{Q}$, until the list is empty $\mathcal{Q} = \{\}$. The procedure of this geometric approach is summarized in Algorithm 3 and illustrated in Figure 4.5.

The main advantage of this approach, compared to the aforementioned method (Baotić, 2002), is that here we have theoretical guarantee of covering the entire set of all feasible parameters $\Omega = \cup_j \mathcal{R}_j$. On the other hand, the computational complexity is increased, since (mainly) the set difference is computationally expensive technique. Another drawback of this approach is that if problem (4.42) is feasible with $\epsilon = 0$, hence the investigated critical region would be degenerated, an additive handling is needed to be performed (e.g. perturbation) in order to find a new, suitable parameter $\theta_0 \in \Omega$. This issue is actually commonly occurring in this method. Furthermore, additional post-processing is required, since this method can create duplicative regions. For example in Figure 4.5(d) the critical region $\mathcal{R}_2$ is constructed, which however will be refunded when the subset $\mathcal{Q}_2$ will be explored. Therefore, it is recommended to keep list of active constraints for all critical regions. Then, we can simply remove regions, which have the same active constraints. However, even this technique is insufficient, because two same regions do not need to have identical active sets, what occurs due to redundant constraints.

Algorithm 3: Algorithm of (Bemporad et al., 2002b).

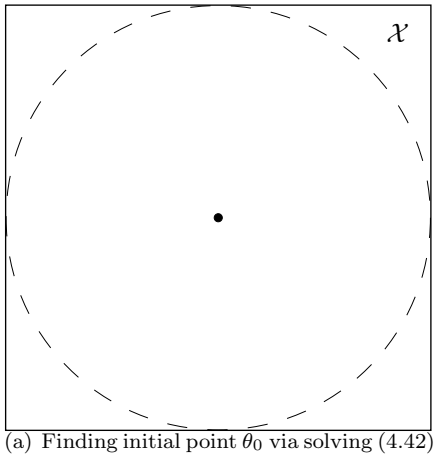
Input: $\mathcal{X}$

Output: Ω, $J^\star(\theta)$, $\mu^\star(\theta)$, $\forall \theta \in \Omega$

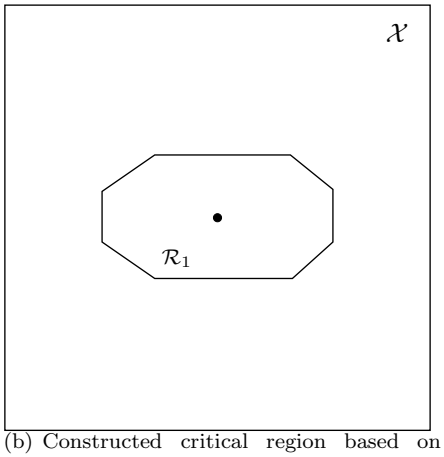
Initialization: $\mathcal{Q} \leftarrow \mathcal{X}$, $j \leftarrow 0$;
while $\mathcal{Q} \neq \{\}$ **do**
  Extract the first candidate $\mathcal{Q}_0 \leftarrow \mathcal{Q}_1$;
  Solve (4.42) for $\mathcal{Q}_0$ to obtain $\theta_0 \in \Omega$;
  if $\epsilon \leq 0$ **then**
    Remove the investigated candidate: $\mathcal{Q} \leftarrow \mathcal{Q} \backslash \mathcal{Q}_0$;
  else
    Find active set constraints:
    $\mathcal{A}_{\theta_0} \leftarrow \{i \in \mathcal{I} \mid G_i z^\star(\theta_0) - w_i - E_i \theta_0 = 0\}$;
    Update increment: $j \leftarrow j + 1$;
    Synthesize: $\mu_j^\star(\theta), J_j^\star(\theta)$ and $\mathcal{R}_j$;
    Set difference: $\mathcal{Q}_{n_\mathcal{Q}} \leftarrow \mathcal{Q}_0 \backslash \mathcal{R}_j$, $n_\mathcal{Q} = 1, \ldots, k$;
    Remove examined candidate: $\mathcal{Q} \leftarrow \mathcal{Q} \backslash \mathcal{Q}_0$;
    Add new candidates: $\mathcal{Q} \leftarrow \mathcal{Q} \cup_{n_\mathcal{Q}} \mathcal{Q}_{n_\mathcal{Q}}$
  end
end
return $\Omega \leftarrow \cup_j \mathcal{R}_j, \mu^\star(\theta) \leftarrow \{\mu_j^\star(\theta)\}, J^\star(\theta) \leftarrow \{J_j^\star(\theta)\}$;

Remark 4.2.4 *As mentioned in Remark 2.2.18, the extended Chebychev ball problem in* (4.42) *can actually write a ball multiple times into the investigated critical region $\mathcal{Q}_0$, with different centers θ_0, yet with the same radius ϵ. But this fact has only a minor impact to this geometric approach, since in the end all critical regions will be found. The only difference is that they might have different indexes j. However, due to the commutative law, we have that entire Ω will be covered.*

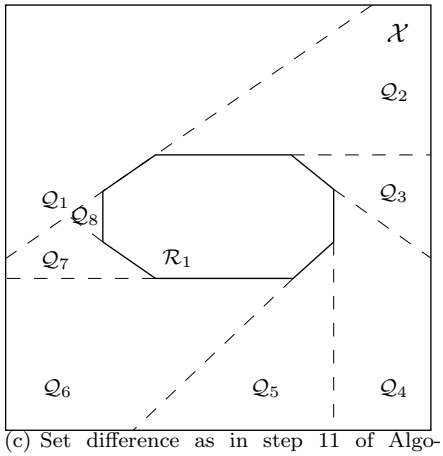
Remark 4.2.5 *Note that including constraints from the set difference operation into matrix G in* (4.3) *(the 8-th step of Algorithm 3) can be counterproductive, since as it is depicted in Figure 4.5(d) we would end up with two critical regions instead of one. (First one would lay in the subset $\mathcal{Q}_1$, while the other part in $\mathcal{Q}_2$.)*



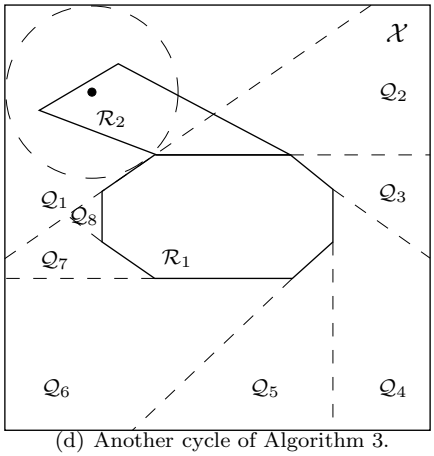
(a) Finding initial point θ_0 via solving (4.42).



(b) Constructed critical region based on given θ_0.



(c) Set difference as in step 11 of Algorithm 3.



(d) Another cycle of Algorithm 3.

Figure 4.5: Figure illustrates the Algorithm 3. Here the black dot represent the center of Chebychev ball (black-dashed circle) θ_0 obtained from LP (4.42), based on which critical regions $\mathcal{R}_j$ is constructed.

4.3 Online Implementation of Explicit MPC

As was shown in Chapter 3, MPC is widely adopted control strategy, due to its natural advantages that significantly outstrips its rivals such as PID, LQR,... On the othe hand, MPC has its disadvantages as well. From the Table 3.1, where these properties are compactly summarized, one can see that the most evident drawback

of this methodology is in the implementation cost. This is caused mainly by the computational burden, since in order to solve a MPC optimization problem requires a powerful computational hardware and an appropriate solver. Therefore, in this section we will aim at this deficiency and the contribution of explicit approach of MPC in this field.

Implicit MPC is commonly implemented in receding horizon fashion (illustrated in Figure 3.1), where entire optimization problem has to be solved every sample instance and only first control input is applied to the plant. The online implementation of implicit MPC can be summarized by

1. Obtain (estimate) current state $x \in \mathbb{R}^n$ and update parameter $\theta \in \mathbb{R}^{n_\theta}$.
2. Solve (4.1) with θ to obtain $\mu^\star \in \mathbb{R}^{n_z}$.
3. Implement $\mu^\star$ into plant ($\kappa \subseteq \mu^\star$ respectively).
4. Repeat from step 1 at the next sampling instant.

It is evident that the computational burden of this implicit MPC approach is in the second step of the algorithm. This is mainly due to fact, that it has to be computed in the aforementioned industrial control platforms, which computational potential is strictly limited. Furthermore, if this computational effort wants to be accelerated, then one need to purchase a new hardware or (and) a modern commercial solver. Since both of these options lead to a major financial investment, companies normally are trying to find a different option.

However, these shortcomings were abolished with the established explicit MPC approach, which offered new opportunities in implementation. This method has successfully reduced the computation burden (of the second step of the algorithm) via offline computing the whole optimization problem (4.1) for all parameters θ by using a multiparametric programming. The procedure, as well as the results of this programming, have been already discussed and analyzed in Section 4.1. Yet we should note that this offline computation can be performed in any computing device (even e.g. a commercial server) and not only the industrial ones, where implicit MPC performs all computation. The implementation algorithm of explicit MPC can be summarized by these steps:

1. For a given problem (4.1) offline compute optimizer $\mu^\star(\theta)$, objective function $J^\star(\theta)$ and polytopic partition Ω.

2. Obtain (estimate) current state $x \in \mathbb{R}^n$ and update parameter $\theta \in \mathbb{R}^{n_\theta}$.

3. Obtain $\mu^\star$ from $\mu^\star(\theta)$.

4. Implement $\mu^\star$ into plant ($\kappa \subseteq \mu^\star$ respectively).

5. Repeat procedure from step 2.

The main difference, compared to implicit implementation, is that the optimization problem is computed only once and for all feasible parameters. Subsequently, as the actual parameter θ is obtained, one need to compute $\mu^\star$, that will be sent back to the plant, hence to achieve a feedback. This method is referred as a Point Location (PL) problem and it represents the most time consuming operation in the online implementation of explicit MPC.

4.3.1 Point Location Problem

The point location problem (Snoeyink, 1997) can be interpreted as a technique that for a given state measurement $\theta \in \mathbb{R}^{n_\theta}$ assigns optimal control inputs $\mu^\star \in \mathbb{R}^{n_z}$. In order to achieve this goal it is required to have a solution of multiparametric programming in hand. The construction, as well as their properties, have been analyzed in Section 4.1. It is well known that optimizer $\mu^\star(\theta)$ is a polytopic PWA function defined over polytopic partition Ω, which is composed of $i = 1, \dots, M$ regions $\Omega = \cup_i \mathcal{R}_i$. Hence we have M different affine control laws, from which we need to assign $\mu^\star$. The question here arises, which of these M control laws should be applied for a given parameter (state measurement) θ. Generally the PL problem consists of two main tasks:

Task 1: Find the index i, of the active region $\mathcal{R}_i$, based on the current value of parameter θ, such that $\theta \in \mathcal{R}_i$.

Task 2: Extract optimal control inputs $\mu^\star$ from the control law $\mu^\star(\theta)$ that are associated with index i.

The second operation of PL problem is computationally inexpensive, since once the index i of active region $\mathcal{R}_i$ is obtained, the optimal control inputs $\mu^\star$ are then determined only via one affine equation

$$\mu^\star = F_{z,i}\theta + g_{z,i}, \tag{4.43}$$

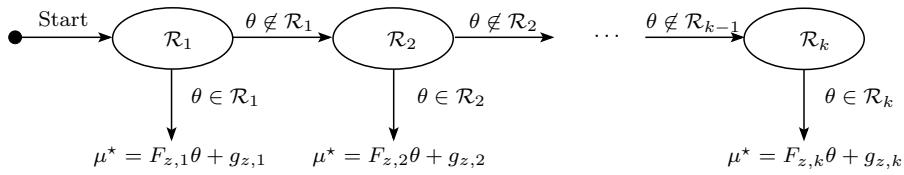


Figure 4.6: Point location problem via sequential search.

what involves only a simple mathematical operations, i.e. multiplication and a sum of two matrices $F_{z,i} \in \mathbb{R}^{n_z \times n_\theta}$ and $g_{z,i} \in \mathbb{R}^{n_z}$. On the other hand, the first operation is crucial, because basically the whole computation effort of the PL problem is here involved. In sequel we will introduce two basic PL algorithms followed be a short discussion.

Sequential Search

PL problem based on the sequential search represents the basic and most straightforward approach in this field. The main idea of this approach is to lists through all regions in order to find out whether the current parameter (state measurement) lies in it $\theta \in \mathcal{R}_i$. Since we have that $\theta \in \Omega$ and $\Omega = \cup_i \mathcal{R}_i$, it is obvious that eventually we will across to a region $\mathcal{R}_i$, which will satisfy such criteria. Sequential search technique is summarized in Algorithm 4 and illustrated in the Figure 4.6.

Algorithm 4: Point location problem via sequential search.

Input: θ
Output: $\mu^\star$

```
for j = 1, ..., M do
    if θ ∈ R_j then
        i ← j;
        break;
    end
end
return μ* ← F_{z,i} θ + g_{z,i};
```

From the computational point of view, this algorithm scales linearly with the number of regions. E.g. denote t_c the time required to verify whether $\theta \in \mathcal{R}_j$ by comparing all its half-spaces, we have that the entire computational time T_c of the Algorithm 4 can be expressed as a linear function $T_c = t_a + \sum_{j=1}^{i} t_c$, where i is the

index of the active region and t_a time required to extract the optimal control input $z^\star$. Since T_c may vary on the basis of actual active index i, the more appropriate information of the required time for PL problem is hence the worst case, where we need to go through each region. Therefore, the worst computational time of PL problem, based on sequential search, is given by $T_c = t_a + Mt_c$, where M is the total number of regions that covers entire feasible set of parameters Ω. It is evident that the number of regions M plays here a major role, since the larger M gives a greater chance to jeopardize the condition $T_c \leq T_s$, which basically says, that the computation time T_c (required to compute the control input) must be shorter or equal as the sampling time T_s of the system.

Remark 4.3.1 *We should note that a in sequential search is prone to continuity of the optimizer, because e.g. for a discontinuous control law $\mu^\star(\theta)$ as in* (4.33) *and a parameter (state measurement) $\theta \in \mathcal{R}_j \cup \mathcal{R}_{j+1}$ we can obtain two different control inputs $\mu_j^\star \in \mathcal{R}_j$ and $\mu_{j+1}^\star \in \mathcal{R}_{j+1}$. Then, based on Algorithm 4, the control input $\mu^\star \in \mathbb{R}^{n_\theta}$, which will be sent into controlled system, is identified in respect to the index j (since $j < (j+1)$), thus we have that $\mu^\star = \mu_j^\star = F_{z,j}\theta + g_{z,j}$. However, there is no guarantee that such selected control input is optimal, in a sense that it do not necessarily possess the lowest value of objective function $J^\star(\theta)$. Therefore, we are forced to check whether $J_j^\star(\theta) \leq J_{j+1}^\star(\theta)$ holds, where $J_j^\star(\theta)$ is associated with $\mu_j^\star$ and $J_{j+1}^\star(\theta)$ with $\mu_{j+1}^\star$. The consequence of this verification is reflected in an additional computational effort required to obtain optimal control input $\mu^\star$ and, what is worst, an additional memory requirement subjected to storing objective function $J^\star(\theta)$. On the other hand, e.g. in the PL approach presented in Algorithm 5, this problem is circumvented.*

PL problem via exploiting convexity of the objective function

As it was suggested in Borrelli et al. (2001) the active index i can be obtained by exploiting convexity of a PWA value function $J^\star(\theta)$. Here, the first operation of PL problem is proceeded by evaluation of all affine functions of $J^\star(\theta)$ for a given parameter (state measurement) θ. Then the maximum value among them is found that is associated with a certain index i. Subsequently, since active index is known, we can compute the optimal control inputs via appropriate control affine expression $\mu^\star = F_{z,i}\theta + g_{z,i}$. The overall procedure of this approach can be summarized by Algorithm 5 and illustrated in Figure 4.7, where a PWA objective function

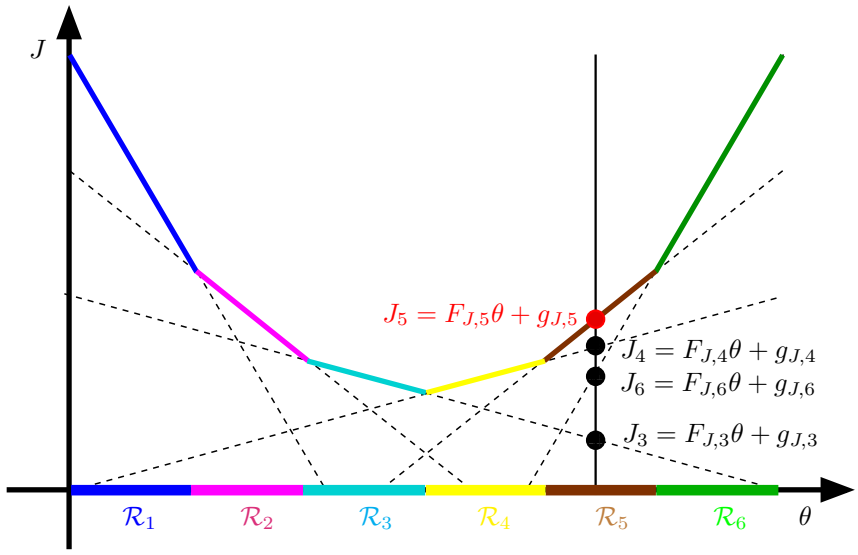


Figure 4.7: Point location problem via exploiting convexity of a PWA value function.

is considered defined over six regions and the actual value of the parameter θ is denoted by the black solid line.

Algorithm 5: Point location problem via exploiting convexity of a PWA value function.

Input: θ

Output: $\mu^\star$

for $j = 1, \ldots, M$ **do**
  $J_i \leftarrow F_{J,i}\theta + g_{J,i}$;
end
$i \leftarrow max(J_i)$;
return $\mu^\star \leftarrow F_{z,i}\theta + g_{z,i}$;

This approach not only accelerates the evaluation speed of PL problem, but at the same time it also reduces the online storage demands of explicit solution, since the Algorithm 5 does not require information about polytopic partition Ω. Compared to the sequential search, this technique has fixed evaluation time. $T_c = t_a + M(t_J + t_m)$, where t_a time required to extract the optimal control input $z^\star \in \mathbb{R}^{n_z}$, t_J is time needed to evaluate all objective values for the given parameter $\theta \in$

$\mathbb{R}^{n_\theta}$ and t_m time associated with computing maximum value. The computational and memory requirements, of these two methods (for both LP/QP), are compared in (Baotić et al., 2008).

Remark 4.3.2 *It should be noted, that the efficient implementation of explicit MPC algorithm based on exploiting the convexity of the objective function, proposed in Borrelli et al. (2001), is not yet directly applicable. This problem manifests from the fact that the convex objective function $J^\star(\theta)$ may posses two identical expressions in two different regions, while the control laws $\mu^\star(\theta)$ above these regions are different. In another words, consider solution of mp-LP given as in* (4.34)*,* (4.32) *and* (4.33)*, respectively, then $(\mathcal{R}_i, \mathcal{R}_j)$, $i \neq j$, $(F_{J,i}, g_{J,i}) = (F_{J,j}, g_{J,j})$, $(F_{z,i}, g_{z,i}) \neq (F_{z,j}, g_{z,j})$. Subsequently the aforementioned algorithm will not be able to apply as it might lead to incorrect (non-optimal) control inputs. This obstacle was, however, overcame in (Anh, 2015, Section 7.1) where, by means of convex lifting, a new (convex and continuous) objective function was constructed such that there was defined an unique expression for each region.*

Further readings and discussions

In Bemporad et al. (2002b) it was pointed out that one can construct a binary tree while the mp-QP problem is solved offline. Unfortunately, here it was not quite clear how to adjust the algorithm into more appropriate form, such that the search tree would be balanced. This deficiency has been however removed in Tøndel et al. (2002), where authors proposed primary to minimize the time required in PL problem and secondary the data storage via more efficient data structure of the explicit solution. The main contribution of this technique lies in the PL acceleration in a sense that its evaluation time is in the best scenario only logarithmic in the number of regions M. However, in the worst scenario it is linear in the total number of facets, what relegates this method (in the worst case) on the same level as is the previous sequential search summarized in Algorithm 4. Even though, that the number of facets is larger then the number of regions, this method works fine for a wide range of smaller, less complex problems, respectively. Numerous authors have made a great effort and further investigated in this field (see e.g. Christophersen et al. (2007); Herceg et al. (2013b); Johansen and Grancharova (2003)).

Remark 4.3.3 *It is worth to emphasize that all PL problems require only a few lines in the control script. Thus their certification is not so costly as in the implicit*

implementation of MPC. Moreover, they can be easily implemented in different programming languages.

Part II

Theoretical Contributions

Chapter 5

Memory Reduction in Explicit Model Predictive Control

Explicit model predictive control is a very interesting control strategy, especially when one aims at implementation of MPC in a fast and computational less demanding fashion. The main contribution of explicit MPC lies in the fact that the online computation is shifted offline, where the multiparametric programming is exploit in order to obtain an optimizer as a function of state parameters, which takes a form of PWA function defined over polytopic partition (see Section 4.1). It can be also noted that such computation can be performed in any computational device (e.g. commercial servers). The online computational effort is then reduced primarily to a point location problem (described in Section 4.3.1) that requires only mere function evaluation, which can be carried out efficiently even on a hardware with low computational resources. This has opened a new possibilities of implementation of MPC even to processes with fast dynamics, where the traditional implicit MPC implementation was impracticable due to rapid sampling rates or software reliability issues.

On the other hand, these advantages of explicit MPC comes with new limitations that can be separated into two categories:

Offline computation demands

This restriction is associated with the computational burden in the offline part of explicit MPC, hence in the multiparametric programming. In general

such an obstacle prevents the widespread use of explicit MPC for systems of moderate or higher complexity. Even though that this offline computation can be performed even on off-the-shell devices, such personal computers or servers, this limitation is reached when one aims at a process which has number of states and control inputs greater then five. However, in the presence of new modern techniques (cf. Section 4.2), the practical use of explicit MPC has been extended even to systems, which optimization problems do not exceed five optimized variables, while the number of parameters (states) can easily go over fifty.

Hardware limitation

As shown in Section 4.1.4 the solution of multiparametric algorithm is an optimizer that is encoded as a PWA function defined over certain polytopic partition. The problem however arises from the fact that the complexity of such explicit optimizers, in a terms of number of region over which this polytopic function is defined, grows exponentially with the prediction horizon, the number of optimized variables, respectively (see e.g. Borrelli (2003b); Grieder (2004)). Therefore, the required memory footprint of such solution can be prohibitive for a large scale of commonly used industrial hardware, since their storage capacities are limited only up to a several kilobytes.

In this chapter, we will deal with the second limitation of explicit MPC. More specifically, we will propose a novel technique which reduces the memory footprint of explicit optimizers by decreasing the number of regions over which this control policy is defined. The method assumes that the complex explicit MPC feedback law $\mu(x)$ is given and the objective is to replace it by a new PWA function $\widetilde{u}(\cdot)$ such that it will provide recursive satisfaction of the original constraints and asymptotic closed-loop stability, while minimizing the integrated square error between $\mu(\cdot)$ and $\widetilde{u}(\cdot)$ (hence mitigating the suboptimality). By following his procedure we will obtain a new explicit feedback law $\widetilde{u}(\cdot)$, which is safe (i.e., it provides constraint satisfaction and closed-loop stability), and is nearly optimal.

In the sequel we start by a literature overview of the proposed technique, followed by the problem statement and its subsequent solution. The efficiency of the proposed method will be demonstrated on three illustrative examples.

5.1 Comprehensive Overview of Proposed Complexity Reduction Techniques

The natural advantages of explicit MPC implementation has attracted a lot of interested researchers, which have put a lot of effort to keep the complexity of explicit MPC solutions on the admissible level. Therefore, numerous techniques have been developed to decrease the required memory storage capacity, in order to meet the required limitations on the targeted control platform. Nowadays, there is wide variety of these techniques, which offers one to choose the most suitable one for the given application. Moreover, one can even combine them by means of applying different technique on each reduction level. It should be, however, noted that each method has usually its price, which can be expressed by e.g. computational burden of a technique, suboptimality of the resulting optimizer, or increased computational time in the point location problem. Nevertheless, by putting besides their differences, their common goal is still the same and can be referred as *complexity reduction* in explicit MPC and in general there are two principal directions presented in the literature.

Optimal complexity reduction

This reduction technique aims at replacing the original complex control law by a new simpler one, such that the optimality of the controller is retained. This goal can be achieved e.g. by employing optimal region merging (ORM) that was proposed in Geyer et al. (2004). ORM is based on merging the regions, which affine expressions are identical and whose union is a convex set (larger region). The disadvantage of this technique lies in the computational burden, since such merging is a NP-hard problem. This makes ORM prohibitive in greater (more complex) explicit optimizers. However, as was shown in Kvasnica and Fikar (2012), the saturated regions can be simply removed and subsequently the blank spaces can be fixed via using so-called clipping filter, which ensures that entire set of feasible parameters is covered. As it was already mentioned in Baotić et al. (2008) authors have shown that the point location problem can be carried out based on the convexity of objective function, thus without need of storing polytopic partition. On the other hand authors in Borrelli et al. (2010) and in Kvasnica et al. (2015) respectively have shown that the polytopic partition does not need to be

computed at all, instead it was advised to store less memory demanding dual optimizers. Furthermore, one can exploit a lattice representation of PWA function (see Wen et al. (2009)) or inner and outer approximations (Oravec et al., 2013) to achieve the goal. The memory footprint of explicit solution can be decreased not only by reduction of the total number of regions, but also by a technique which aims to the problem from a different perspective. For example, in Sz

Hucs et al. (2011) a three layer compression technique was shown, where the data of explicit MPC solution are shrinked to a fraction of the original size proportion. On the other hand, such compressed data has to be decripted at each sampling instance of the online procedure, hence an additional computational effort is needed. This technique is useful when one aims at control device with acute shortage of available memory, but sufficient computational power.

Suboptimal complexity reduction

Function approximation technique tries to find a new explicit optimizer, such that the complexity of such fitted function is reduced, while the optimality is allowed to be reduced. Hence we can say that the complexity of PWA control law is being exchanged for the suboptimality. Numerous methods have been proposed e.g. that exploits freedom of Lyapunov function (Lu et al., 2011), move-blocking (Cagienard et al., 2007a), polynomial approximations (Kvasnica et al., 2011; Valencia-Palomo and Rossiter, 2010) inner and outer approximations (Jones and Morari, 2010), approximation of PWA control laws that are defined over simplical (Bemporad et al., 2011) or hypercube (Johansen and Grancharova, 2003) domains, bounded PWA approximation of the cost function (Holaza et al., 2012), and many others. It should be noted, that methods that sacrifice the performance usually achieve better memory reduction, compared to methods that maintain optimality. Moreover, the level of the suboptimality is typically proportional to the complexity reduction.

5.2 Notation and Problem Statement

5.2.1 Notation

In this entire section, for simplicity, we are considering MPC defined as a QP with control to the origin. Therefore, we denote the state vector to be identical with the vector of parameters, i.e. $x = \theta \in \mathbb{R}^n$, and the vector of inputs to be the vector of all optimization variables, i.e. $u = z \in \mathbb{R}^m$. Next, let us recall that the x is the state measurement (parameter) and x_k to be the k-th prediciton of the state at the time t. Moreover, denote vert($\mathcal{P}$) vertices of a polyhedron $\mathcal{P}$, $triangulate(\cdot)$ the triangulation technique (cf. Definition 2.2.17), $\text{proj}_x(\cdot)$ the projections into x-space (cf. Definition 2.2.16) and diag($\cdot$) to be a diagonal matrix.

5.2.2 Problem Statement

We consider the control of linear discrete-time systems in the state-space form (3.5), subjected to polytopic constraints (3.6). We are interested in obtaining a feedback law $\mu : \mathbb{R}^n \to \mathbb{R}^m$ which drives all states of (3.5) to the origin while providing recursive satisfaction of state and input constraints, i.e., $\forall t \in \mathbb{N}\ x(t) \in \mathcal{X}$, $u(t) \in \mathcal{U}$. For such setup, we have constructed the following optimization problem:

$$\mu = \arg\min \sum_{k=0}^{N-1} (x_{k+1}^T Q_x x_{k+1} + u_k^T Q_u u_k) \tag{5.1a}$$

$$\text{s.t. } x_{k+1} = Ax_k + Bu_k,\ k = 0, \dots, N-1, \tag{5.1b}$$

$$u_k \in \mathcal{U},\ k = 0, \dots, N-1, \tag{5.1c}$$

$$x_k \in \mathcal{X},\ k = 0, \dots, N-1, \tag{5.1d}$$

$$x_0 = x(t). \tag{5.1e}$$

As it was shown in Section 4.1, by solving (5.1) for all feasible initial conditions via multiparametric programming, per Theorem 4.1.8, one obtains explicit optimizer of the form

$$\mu(x) = F_j x + g_j \text{ if } x \in \mathcal{R}_j,\ j = 1, \dots, M, \tag{5.2}$$

where $F \in \mathbb{R}^{m \times n}$, $g \in \mathbb{R}^m$ are gains of the appropriate affine control law and $\Omega = \cup_j \mathcal{R}_j = \text{dom}(\mu(\cdot))$ being the polytopic partition.

The main problem of explicit MPC is that the complexity of the feedback law $\mu(x)$ in (5.2), expressed by the number of polytopes M, grows exponentially with

the prediction horizon N. Moreover, this issue comes hand in hand with the online computation complexity (see PL problem in Section 4.3.1). In another words, the more polytopes constitute $\mu(x)$, the more memory is required to store the function in the control hardware and the longer it takes to obtain value of the optimizer for a particular value of the state measurements. Therefore, in order to satisfy the hardware limitations, hence to guarantee a successful implementation into control platforms, we want to replace $\mu(x)$ by a similar, yet less complex, PWA feedback law $\widetilde{u}(x)$. We propose to perform this substitution such that the approximated optimizer $\widetilde{u}(x)$ will preserve recursive satisfaction of original constraints (5.1c) and (5.1d). The price that we are willing to pay, in favor of achieving smaller complexity, can be expressed in terms of increased suboptimality of $\widetilde{u}(x)$ with respect to the optimal representation $\mu(x)$. Therefore, our approach can be categorized as suboptimal complexity reduction technique.

Problem 5.2.1 *Assume $\mu : \mathbb{R}^n \to \mathbb{R}^m$ to be a PWA explicit representation of the MPC feedback law defined as in* (5.2)*. Our goal is to construct a new PWA optimizer $\widetilde{u} : \mathbb{R}^n \to \mathbb{R}^m$ as*

$$\widetilde{u}(x) = \widetilde{F}_i x + \widetilde{g}_i \text{ if } x \in \widetilde{\mathcal{R}}_i,\ i = 1, \dots, \widetilde{M}, \tag{5.3}$$

with matrices $\widetilde{F} \in \mathbb{R}^{m \times n}$ and $\widetilde{g} \in \mathbb{R}^m$, where $\widetilde{M} < M$. Moreover, we propose to synthesize $\widetilde{u}(x)$ such that:

R1: the recursive satisfaction of state and input constraints as in (3.3d) *is preserved, thus for all $t \in \mathbb{N}$ we have that $\widetilde{u}(x) \in \mathcal{U}$ and $Ax + B\widetilde{u}(x) \in \mathcal{X}$;*

R2: the asymptotic stability of the closed-loop system

$$x(t+1) = Ax(t) + B\widetilde{u}(x(t)) \tag{5.4}$$

is provided with respect to the origin as an equilibrium point;

R3: the squared error between $\widetilde{u}(x)$ and $\mu(x)$ on the entire domain Ω is minimized via exploiting integral

$$\min \int_{\Omega} \|\mu(x) - \widetilde{u}(x)\|_2^2 \, dx, \tag{5.5}$$

where dx denotes the Lebesgue measure of Ω, see e.g. Baldoni et al. (2010);

The Problem 5.2.1 is illustrated in the Figure 5.1. Here a new optimizer $\widetilde{u}(x)$, drawn by red color, is constructed based on the original feedback $\mu(x)$, shown in black, via minimization of integral of the squared difference between them (5.5). The reduced complexity is here evident, since $\mu(x)$ is defined over 7 regions, while $\widetilde{u}(x)$ is defined only over 3 regions. Yet, the cost of this reduction is expressed in terms of optimality, what will be demonstrated in case studies.

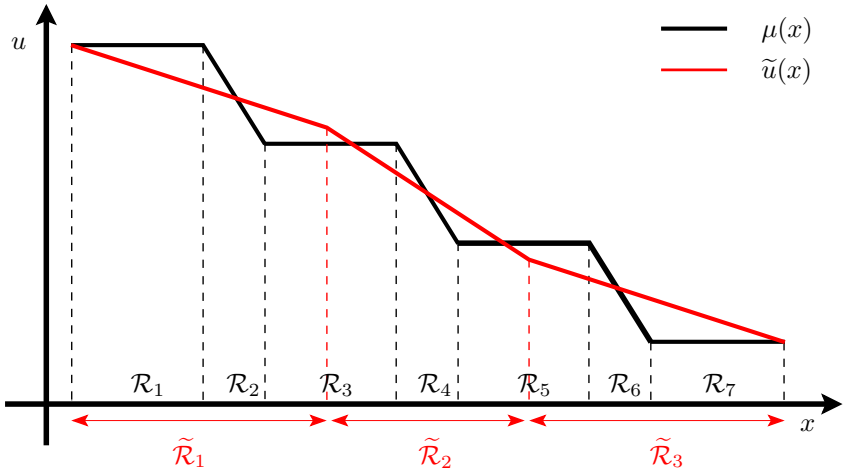


Figure 5.1: Illustrative example of problem statement.

Since the aforementioned Problem 5.2.1 is not an easy task, we propose to solve it via three steps. In the first step we will construct a simple polytopic partition $\widetilde{\Omega}$ that will be composed of a fewer number of regions $\cup_i \widetilde{\mathcal{R}}_i = \widetilde{\Omega}$ for all $i = 1, \dots, \widetilde{M}$, such that $\widetilde{M} < M$, where M denotes the number of regions associated with the partition Ω of the optimal feedback $\mu(x)$. Moreover, in order to provide criterion R1 we have that the union of new regions must be equal to union of the original regions

$$\bigcup_i \widetilde{\mathcal{R}}_i = \widetilde{\Omega} = \Omega = \bigcup_j \mathcal{R}_j, \tag{5.6}$$

hence we will have guarantee that domains of both optimizers $\widetilde{u}(x)$ and $\mu(x)$ will be identical. Subsequently, if such domain $\widetilde{\Omega}$ will be known, we move to the second step. Here parameters $\widetilde{F}$ and $\widetilde{g}$ of the optimizer $\widetilde{u}(x)$ are optimized such that the error between $\mu(x)$ and $\widetilde{u}(x)$ in (5.5) is mitigated and all of the original constraints as in (3.3d) are preserved. In the last third step we will provide a closed-loop stability criterion of R2.

5.3 Constrution of the Polytopic Partition

The objective of this section is to construct a new (less complex) polytopic partition $\widetilde{\Omega} = \cup_i \widetilde{\mathcal{R}}_i$ with $i = 1, \ldots, \widetilde{M}$, such that

$$\widetilde{\Omega} = \Omega, \tag{5.7a}$$

$$\widetilde{M} < M, \tag{5.7b}$$

will be satisfied. By recalling from Section 4, the complexity of explicit optimizers, in a terms of number of regions over which they are defined, grows (in the worst case) exponentially with the prediction horizon N. Hence, to satisfy (5.7b), we propose to obtain the polytopic partition $\widetilde{\Omega}$ by solving the same optimization problem (5.1) once again, but with a shorter prediction horizon $\hat{N} < N$. Then, by Theorem 4.1.8, we obtain the feedback law $\widehat{\mu}(\cdot)$ as a PWA function of x:

$$\widehat{\mu}(x) = \hat{F}_i x + \hat{g}_i \text{ if } x \in \widetilde{\mathcal{R}}_i,\ i = 1, \ldots, \widetilde{M}, \tag{5.8}$$

which is defined over $\widetilde{M}$ polytopes $\widetilde{\mathcal{R}}_i$. However, it is well known that with $\hat{N} < N$ also the volume of the polytopic partition might decrease, i.e. $\text{vol}(\widetilde{\Omega}) \subseteq \text{vol}(\Omega)$, what is in a conflict with condition (5.7a). To tackle this problem, we suggest to replace the original state constraint $\mathcal{X}$ in(5.1) by a new unique one, where the maximal control invariant set will be employed.

Definition 5.3.1 (Maximum control invariant set) *Let $x_{k+1} = Ax_k + Bu_k$ be a linear system that is subject to constraints $x \in \mathcal{X} \subseteq \mathbb{R}^n$, $u \in \mathcal{U} \subseteq \mathbb{R}^m$. Then the set*

$$\mathcal{C}_\infty = \{x_0 \in \mathcal{X} \mid \forall k \in \mathbb{N} :\ \exists u_k \in \mathcal{U} \text{ s.t. } Ax_k + Bu_k \in \mathcal{X}\} \tag{5.9}$$

is called the maximum control invariant set.

Assuming that system dynamics is described by $x_{k+1} = Ax_k + Bu_k$ and constraints $\mathcal{X}, \mathcal{U}$ and $\mathcal{F}$ are polytopes denoted as in (2.7) with an appropriate index, i.e.

$$\mathcal{X} = \{x \in \mathbb{R}^n \mid A_x x \leq b_x\},$$
$$\mathcal{U} = \{u \in \mathbb{R}^m \mid A_u u \leq b_u\},$$
$$\mathcal{F} = \{x \in \mathbb{R}^n \mid A_F x \leq b_F\},$$

then the procedure of computing maximum control invariant set can be summarized as in Algorithm 6.

Algorithm 6: Maximum control invariant set $\mathcal{C}_\infty$.

Input: $\mathcal{X}$,$\mathcal{U}$, A, B

Output: $\mathcal{C}_\infty$

Initialization: $\mathcal{F}_0 \leftarrow \mathcal{X}$, $k = 0$;

repeat

$k \leftarrow k + 1$;

$$\mathcal{F}_k \leftarrow \text{proj}_x \left(\begin{bmatrix} A_F A & A_F B \\ A_F & \mathbf{0} \\ \mathbf{0} & A_u \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} \leq \begin{bmatrix} b_F \\ b_F \\ b_u \end{bmatrix} \right);$$

until $\mathcal{F}_k = \mathcal{F}_{k-1}$;

return $\mathcal{C}_\infty \leftarrow \mathcal{F}_k$;

Remark 5.3.2 *The Algorithm 6 terminates in a finite iteration if the system is open-loop stable and $\mathcal{X}$ is bounded and contains the origin. Otherwise one should add a condition that terminates Algorithm 6 in a finite number of iterations (see e.g. Gilbert and Tan (1991)).*

By exploiting (5.9) into (5.1) one can obtain optimization problem

$$u = \arg\min \sum_{k=0}^{N-1} (x_{k+1}^T Q_x x_{k+1} + u_k^T Q_u u_k) \tag{5.11a}$$

$$\text{s.t. } x_{k+1} = Ax_k + Bu_k, \ \forall k = 0, \ldots, N-1, \tag{5.11b}$$

$$u_k \in \mathcal{U}, \ \forall k = 0, \ldots, N-1, \tag{5.11c}$$

$$x_0 \in \mathcal{C}_\infty, \ x_1 \in \mathcal{C}_\infty, \tag{5.11d}$$

where $\mathcal{C}_\infty$ is a maximum control invariant set.

Lemma 5.3.3 *Consider $\mu(x)$ as in* (5.2) *to be the solution of* (5.11) *according to Theorem 4.1.8 for some prediction horizon N. Further, let $\widehat{\mu}(x)$ be the explicit MPC feedback function as in* (5.8) *obtained by solving the same optimization problem* (5.11)*, for some prediction horizon $\hat{N} < N$. Then we have that* (5.7a) *holds.*

Proof. As was shown in Algorithm 6 the domain Ω and $\widetilde{\Omega}$ respectively are projections of constraints in (5.1) onto state-space x. Since (5.11d) are the only state constraints of the problem (5.11), Ω is independent of the choice of the predic-

tion horizon. Therefore $\Omega_N = \Omega_{\widehat{N}}$. Finally, since $\cup_j \mathcal{R}_j = \Omega_N = \Omega_{\widehat{N}} = \cup_i \widetilde{\mathcal{R}}_i$ by Theorem 4.1.8, the result follows.

Therefore, the bottom line is, the optimal explicit feedback function $\mu(x)$ as in (5.2) can be obtained by solving (5.11) with some prediction horizon N. Subsequently, by solving the same optimization problem (5.11) for some prediction horizon $\hat{N} < N$, one obtains a simpler optimizer $\widehat{\mu}(x)$ as (5.8) such that recursive satisfaction of the original constraints in (5.1) is provided due to (5.11c) and (5.11d) respectively. Therefore, we have that criterion R1 in Problem 5.2.1 for $\widehat{\mu}(x)$ holds. However, we have no guarantee that $\widehat{\mu}(x)$ minimizes the offset (5.5), thus its performance is questionable. Due to this deficiency, we propose to simply discard this simpler optimizer $\widehat{\mu}(x)$ and preserve only its polytopic partition $\widetilde{\Omega} = \cup_i \widetilde{\mathcal{R}}_i$ for $i = 1, \ldots, \widetilde{M}$, with $\widetilde{M} < M$, since $\hat{N} < N$. That completes the objective (5.7) of this section.

Remark 5.3.4 *The advantage of the procedure presented here is that the domain of $\mu(\cdot)$ is partitioned into $\{\widetilde{\mathcal{R}}_i\}$ in such a way that the approximation problem is always feasible, i.e., there always exist parameters $\widetilde{F}_i$, $\widetilde{g}_i$ in* (5.3) *such that $\widetilde{u}$ guarantees recursive satisfaction of input and state constraints. This is not always the case if an arbitrary partition is selected.*

Remark 5.3.5 *It should be noted that if a smaller prediction horizon $\hat{N}$ is used, then a greater complexity reduction of $\widehat{\mu}(x)$ is achieved. Hence it is recommended to use $\hat{N} \ll N$ e.g. $\hat{N} = 1$. Moreover, to provide the best references of $\mu(x)$ it is advised to use the longest possible prediction horizon N.*

5.4 Function Fitting

In the previous Section 5.3 we have shown how to construct a simple, memory less demanding polytopic partition $\widetilde{\Omega}$ associated with a simpler optimizer $\widetilde{u}(x)$, in respect to the original one Ω belonging to $\mu(x)$, such that (5.7) hold. This was achieved by solving the optimization problem (5.11) for some prediction horizon $\hat{N} < N$, where N denotes prediction horizon used in (5.11) in order to compute the original optimizer $\mu(x)$. However due to the poor performance of this optimizer $\widetilde{u}(x)$, since no offset minimization (5.5) has been provided, only the domain $\widetilde{\Omega}$ was stored.

In this section our goal will be, for the given polytopic partition $\widetilde{\Omega}$, to optimize parameters $\widetilde{F}_i \in \mathbb{R}^{m\times n}$ and $\widetilde{g}_i \in \mathbb{R}^m$ of the approximated optimizer

$$\widetilde{u}(x) = \widetilde{F}_i x + \widetilde{g}_i \text{ if } x \in \widetilde{\mathcal{R}}_i,\ i = 1, \dots, \widetilde{M}, \tag{5.12}$$

such that the error between $\mu(x)$ and $\widetilde{u}(x)$ is minimized and all of the original constraints in (5.1) are preserved. This way both criterion R1 and R3 of Problem 5.2.1 will be satisfied.

Let us recall from Section 4.1.4, the polytopes $\widetilde{\mathcal{R}}_i$ form the partition of the domain of $\widehat{\mu}(x)$, i.e. their respective interiors do not overlap. Therefore we can divide the search for parameters $\widetilde{F}_i$ and $\widetilde{g}_i$ for $i = 1, \dots, \widetilde{M}$ into a series of $\widetilde{M}$ optimization problems formulated as:

$$\min_{\widetilde{F}_i, \widetilde{g}_i} \int_{\widetilde{\mathcal{R}}_i} \|\mu(x) - \widetilde{u}(x)\|_2^2 \, \mathrm{d}x \tag{5.13a}$$

$$\text{s.t. } \widetilde{F}_i x + \widetilde{g}_i \in \mathcal{U},\ \forall x \in \widetilde{\mathcal{R}}_i, \tag{5.13b}$$

$$Ax + B(\widetilde{F}_i x + \widetilde{g}_i) \in \mathcal{C}_\infty,\ \forall x \in \widetilde{\mathcal{R}}_i, \tag{5.13c}$$

where $\mathcal{C}_\infty$ denotes a maximum control invariant set as in Definition 5.3.1. Furthermore constraints (5.13b) and (5.13c) ensures recursive satisfaction of input and state constraints respectively.

However, by solving the optimization problem (5.13), one can encounter with these three technical issues:

1. Even though that only parameter x is restricted to a particular region $x \in \widetilde{\mathcal{R}}_i$, where (5.12) is an affine function, the optimal control feedback $\mu(x)$ may here still attain a PWA character.

2. The integration in (5.13) has to be performed even over polytopes of a greater dimension $n \geq 1$..

3. In order to privide R1 in Problem 5.2.1 both constraints (5.13b) and (5.13c) have to hold $\forall x \in \mathcal{R}_i$, i.e. for an infinite number of points.

In order to circumvent the first limitation, we propose to proceed the intersections of polytopes $\{\widetilde{\mathcal{R}}\}_i$ with $i = 1, \dots, \widetilde{M}$ and $\{\mathcal{R}\}_j$ with $j = 1, \dots, M$. In another words, for a fixed index i of $\widetilde{\mathcal{R}}_i$ we compute $\widetilde{M}$-times expression

$$\mathcal{Q}_{i,j} = \widetilde{\mathcal{R}}_i \cap \mathcal{R}_j,\ \forall j = 1, \dots, M, \tag{5.14}$$

the product of which yields a cell of polytopes $\mathcal{Q}$. Moreover, we have that over a given polytope $\mathcal{Q}_{i,j}$ both explicit MPC feedback functions $\mu(x)$ and $\widetilde{u}(x)$, as in (5.2) and (5.12) respectively, have affine expression. Yet there is another problem that needs to be solved. Specifically the intersection in (5.14) can produce an empty polytope $\mathcal{Q}_{i,j} = \emptyset$. Thus it is convenient to store only indexes of the non-empty intersections as

$$\mathcal{J}_i = \{j = 1, \dots, M \mid \widetilde{\mathcal{R}}_i \cap \mathcal{R}_j \neq \emptyset\}, \tag{5.15}$$

where i is the fixed index of simpler region $\widetilde{\mathcal{R}}_i$.

By applying (5.14) with respect to (5.15) into (5.13a), one can obtain an objective in a form of a sum of integrals evaluated over non-empty intersected polytopes $\mathcal{Q}_{i,j} \in \mathbb{R}^n$ as

$$\min_{\widetilde{F}_i, \widetilde{g}_i} \sum_{j \in \mathcal{J}_i} \int_{\mathcal{Q}_{i,j}} \|(F_j x - g_j) - (\widetilde{F}_i x + \widetilde{g}_i)\|_2^2 \, \mathrm{d}x, \tag{5.16}$$

with known matrices F_j, g_j, associated to complex feedback (5.2), and optimized variables $\widetilde{F}_j$, $\widetilde{g}_j$, associated to approximated feedback (5.12).

In order to obtain an analytic form of the integral in (5.16), we use the results derived in Lasserre and Avrachenkov (2001), which were further extended by Baldoni et al. (2010).

Lemma 5.4.1 (Baldoni et al. (2010)) *Let f be a homogeneous polynomial of degree d in n variables, and let $s_1, \dots, s_{n+1}$ be the vertices of an n-dimensional simplex Δ. Then*

$$\int_\Delta f(y)dy = \gamma \sum_{1 \le i_1 \le \cdots \le i_d \le n+1} \sum_{\epsilon \in \{\pm 1\}^d} \epsilon_1 \cdots \epsilon_d f\left(\textstyle\sum_{k=1}^d \epsilon_k s_{i_k}\right) \tag{5.17}$$

where

$$\gamma = \frac{\mathrm{vol}(\Delta)}{2^d d! \binom{d+n}{d}}, \tag{5.18}$$

and $\mathrm{vol}(\Delta)$ *is the volume of the simplex.*

However, the issue is that Lemma in (5.4.1) is not directly applicable to the objective function (5.16) since the intersected polytopes $\mathcal{Q}_{i,j}$ are not in general simplices. Hence, in order to tackle this problem we propose to tessellate each $\mathcal{Q}_{i,j}$ into simplices via

$$\Delta_{i,j,k} = triangulate(\mathcal{Q}_{i,j}), \tag{5.19}$$

such that

$$\begin{aligned}\mathrm{int}(\Delta_{i,j,k_1}) \cap \mathrm{int}(\Delta_{i,j,k_2}) &= \emptyset,\ \forall k_1 \neq k_2,\\ \cup_k \Delta_{i,j,k} &= \mathcal{Q}_{i,j},\end{aligned}$$

where $k = 1, \ldots, K$ and K denoting the number of simplices created by tessellating polytope $\mathcal{Q}_{i,j}$. Subsequently, we can rewrite (5.16) as a sum of the integrals evaluated over each simplex $\Delta_{i,j,k}$ as

$$\min_{\widetilde{F}_i, \widetilde{g}_i} \sum_{j \in \mathcal{J}_i} \sum_{k=1}^{K_{i,j}} \int_{\Delta_{i,j,k}} \|(F_j x - g_j) - (\widetilde{F}_i x + \widetilde{g}_i)\|_2^2 \, \mathrm{d}x. \tag{5.21}$$

Furthermore, note that Lemma 5.4.1 applies only to homogeneous polynomials. However the function under the integral

$$f(x) := \|(F_j x + g_j) - (\widetilde{F}_i x + \widetilde{g}_i)\|_2^2 \tag{5.22}$$

is not homogeneous. To see this let us expand the quadratic objective function $f(x)$ into

$$f(x) := x^T Q x + r^T x + q, \tag{5.23}$$

where

$$\begin{aligned} Q &= F_j^T F_j - 2F_j \widetilde{F}_i + \widetilde{F}_i^T \widetilde{F}_i,\\ r &= 2(F_j^T \widetilde{g}_i + \widetilde{F}_i^T \widetilde{g}_i - \widetilde{F}_i^T g_j - F_j^T \widetilde{g}_i),\\ q &= g_j^T g_j - 2 g_j^T \widetilde{g}_i + \widetilde{g}_i^T \widetilde{g}_i. \end{aligned}$$

Now it is evident that $f(x)$ is a quadratic function with optimized variables $\widetilde{F}_i$ and $\widetilde{g}_i$, yet is not homogeneous, since not all of its monomials have the same degree (in particular, we have monomials of degrees 2, 1 and 0 in f). However, since an integral is closed under linear combinations, we have that

$$\int_\Delta f(x)\mathrm{d}x = \int_\Delta f_{\mathrm{quad}}(x)\mathrm{d}x + \int_\Delta f_{\mathrm{lin}}(x)\mathrm{d}x + \int_\Delta f_{\mathrm{const}}\mathrm{d}x, \tag{5.25}$$

with $f_{\mathrm{quad}}(x) := x^T Q x$, $f_{\mathrm{lin}} := r^T x$ and $f_{\mathrm{const}} := q$. Since each of these newly defined functions is a homogeneous polynomial of degree 2, 1 and 0, respectively, the integral $\int_\Delta f(x)\mathrm{d}x$ can now be evaluated by applying (5.17) of Lemma 5.4.1 to each integral in the right-hand-side of (5.25). We hence obtain an analytic expression for the integral error as a quadratic function of the unknowns $\widetilde{F}_i$ and $\widetilde{g}_i$.

Remark 5.4.2 *The formulation in* (5.25) *can be simplified, since the degree of the last polynomial in* (5.25) *is equal to* $d = 0$*, from* (5.17) *it follows that*

$$\int_\Delta f_{const} dx = \int_\Delta q dx = q \operatorname{vol}(\Delta),$$

where q *is a constant and* $\operatorname{vol}(\Delta)$ *is denoting a volume of the simplex* Δ*.*

Finally, when optimizing for $\widetilde{F}_i$ and $\widetilde{g}_i$, we need to ensure that the constraints in (5.13) hold for all points $x \in \widetilde{\mathcal{R}}_i$. By our assumptions, the sets $\mathcal{U}$ and $\mathcal{C}_\infty$ are polytopes, hence can be represented by $\mathcal{U} = \{u \,|\, H_\mathrm{u} u \leq h_\mathrm{u}\}$ and $\mathcal{C}_\infty = \{x \,|\, H_\mathrm{c} x \leq h_\mathrm{c}\}$. By using $u = \widetilde{F}_i x + \widetilde{g}_i$, constraint (5.13b) and (5.13c) can be compactly written as

$$\forall x \in \widetilde{\mathcal{R}}_i : f(x) \leq 0, \tag{5.26}$$

with

$$f(x) := \begin{bmatrix} H_\mathrm{u}\widetilde{F}_i \\ H_\mathrm{c}(A + B\widetilde{F}_i) \end{bmatrix} x + \begin{bmatrix} H_\mathrm{u}\widetilde{g}_i - h_\mathrm{u} \\ H_\mathrm{c} B \widetilde{g}_i - h_\mathrm{c} \end{bmatrix}. \tag{5.27}$$

Then we can state our next result.

Lemma 5.4.3 *Let* $\mathcal{V}_i = \{v_{i,1}, \ldots, v_{i,n_{v,i}}\}$*,* $v_{i,j} \in \mathbb{R}^n$ *be the vertices of polytope* $\widetilde{\mathcal{R}}_i$ *(see Definition 2.2.8). Then* (5.26) *is satisfied* $\forall x \in \widetilde{\mathcal{R}}_i$ *if and only if* $f(v_{i,j}) \leq 0$ *holds for all vertices.*

Proof. To simplify exposition, we replace $\widetilde{\mathcal{R}}_i$ by $\mathcal{P}$ to avoid double indexing, and we let the vertices of $\mathcal{P}$ be $v_1, \ldots, v_{n_v}$. As seen from (5.27), $f(\cdot)$ is a linear function of x. Necessity is obvious since $v_j \in \mathcal{P}$ trivially holds for all vertices, cf. Definition 2.2.8. To show sufficiency, represent each point of $\mathcal{P}$ as a convex combination of its vertices v_j, i.e., $z = \sum_j \lambda_j v_j$. Then $f(z) \leq 0 \ \forall z \in \mathcal{P}$ is equivalent to $f(\sum_j \lambda_j v_j) \leq 0$, $\forall \lambda \in \Lambda$, where $\Lambda = \{\lambda \mid \sum_j \lambda_j = 1,\ \lambda_j \geq 0\}$ is the unit simplex. Since $f(\cdot)$ is assumed linear, we have $f(\sum_j \lambda_j v_j) = \sum_j \lambda_j f(v_j)$. Therefore, $\sum_j \lambda_j f(v_j) \leq 0$ holds for an arbitrary $\lambda \in \Lambda$ since $f(v_j) \leq 0$ is assumed to hold, and because each λ_j is non-negative. Therefore $f(v_j) \leq 0 \Rightarrow f(z) \leq 0 \ \forall z \in \mathcal{P}$.

By combining Lemma 5.4.3 with the integration result in (5.25), we can formulate the search for $\widetilde{F}_i$, $\widetilde{g}_i$ from (5.13) as

$$\min_{\widetilde{F}_i, \widetilde{g}_i} \sum_{j \in \mathcal{J}_i} \sum_{k=1}^{K_{i,j}} \int_{\Delta_{i,j,k}} \|(F_j x - g_j) - (\widetilde{F}_i x + \widetilde{g}_i)\|_2^2 \, \mathrm{d}x, \tag{5.28a}$$

$$\text{s.t. } \widetilde{F}_i v_{i,\ell} + \widetilde{g}_i \in \mathcal{U},\ \forall v_{i,\ell} \in \operatorname{vert}(\widetilde{\mathcal{R}}_i), \tag{5.28b}$$

$$A v_{i,\ell} + B(\widetilde{F}_i v_{i,\ell} + \widetilde{g}_i) \in \mathcal{C}_\infty,\ \forall v_{i,\ell} \in \operatorname{vert}(\widetilde{\mathcal{R}}_i), \tag{5.28c}$$

where vert($\widetilde{\mathcal{R}}_i$) enumerates all vertices of the corresponding polytope. Since each polytope $\widetilde{\mathcal{R}}_i$ has only finitely many vertices Ziegler (1994), problem (5.28) has a finite number of constraints. Moreover, the objective in (5.28a) is a quadratic function in the unknowns $\widetilde{F}_i$, $\widetilde{g}_i$ and its analytic form can be obtained via (5.17). Finally, since the sets $\mathcal{U}$ and $\mathcal{C}_\infty$ are assumed to be polytopic, all constraints in (5.28) are linear. Thus problem (5.28) is a quadratic optimization problem for $i = 1, \dots, \widetilde{M}$, where $\widetilde{M}$ is the number of polytopes that constitute the domain of $\widetilde{u}(\cdot)$ in (5.3).

As our next result we show that if polytopes $\widetilde{\mathcal{R}}_i$ are chosen as suggested by Lemma 5.3.3, then (5.28) is always feasible for each $i = 1, \dots, \widetilde{M}$.

Theorem 5.4.4 *Let $\widetilde{\mathcal{R}}_i$, $i = 1, \dots, \widetilde{M}$ be obtained by Lemma 5.3.3 for $\widehat{N} < N$. Then the optimization problem* (5.28) *is always feasible, i.e., for each $i = 1, \dots, \widetilde{M}$ there exist matrices $\widetilde{F}_i$ and vectors $\widetilde{g}_i$ such that the simplified feedback $\widetilde{u}(x)$ from* (5.3) *provides recursive satisfaction of constraints in* (5.1c) *and* (5.1d)*, respectively, for an arbitrary $x \in \Omega$.*

Proof. Obtaining $\widetilde{\mathcal{R}}_i$ via Lemma 5.3.3 amounts to solving the MPC problem (5.1) explicitly for some $\widehat{N} < N$. By doing so, one obtains the explicit controller $\widehat{\mu}(\cdot)$ as in (5.8). Since (5.1) includes the invariant set constraint (5.11d), it follows from Definition 5.3.1 that the feedback $\widehat{\mu}(x) = \widehat{F}_i x + \widehat{g}_i$ provides recursive satisfaction of constraints in (5.1c) and (5.1d), respectively, but is not necessarily optimal with respect to approximating the long-horizon feedback $\mu(\cdot)$. Since the polytopes $\widetilde{\mathcal{R}}_i$ in (5.28) are the same as in (5.8), the choice $\widetilde{F}_i = \widehat{F}_i$ and $\widetilde{g}_i = \widehat{g}_i$ obviously satisfies constraints in (5.28). Finally, since feasibility of (5.28) is independent of the choice of the objective function, the result follows.

Remark 5.4.5 *Optimization problem* (5.28) *naturally covers the multi-input scenario where $\widetilde{F}_i \in \mathbb{R}^{m \times n}$, $\widetilde{g}_i \in \mathbb{R}^m$ with $m \geq 1$.*

Remark 5.4.6 *It is worth mentioning that proposed method* (5.28) *is suitable not only for PWA functions, but also e.g. for PWQ functions. This fact can be exploited e.g. in the dynamic programming (of mp-QP), where one need to approximate cost-to-go function, which attains a PWQ form, to proceed into further iterations (see e.g. (Borrelli et al., 2016)).*

5.4.1 Function Fitting via Vertex Approximation

As was shown in (Holaza et al., 2013) the function fitting as in (5.13) can be approached also via exploiting only vertices. The integral in (5.13a) can be circumvent by a sum

$$\min_{\widetilde{F}_i,\widetilde{g}_i} \sum_{j=1}^{N_w} \|\mu(w_j) - \widetilde{u}(w_j)\|_2^2 \tag{5.29}$$

where $w_i, \ldots, w_{N_w}$ are points which should be used to evaluate the approximation error. The selection of these points can be deducted e.g. from Figure 5.1, since the only points where either optimizer $\mu(x)$ or $\widetilde{u}(x)$ change the tangents of the respective local affine functions are in the vertices of corresponding regions. However, as it has been already discussed at the beginning of this section (first issue), for a general n-dimensional scenario with $n > 1$ we need to consider the points from the intersections (5.14), more specifically only the non-empty ones (5.15). Thus, the approximation points in (5.29) are given as

$$\mathcal{W} = \{\text{vert}(\mathcal{Q}_{i,j}) \mid \mathcal{Q}_{i,j} \neq \emptyset\}, \tag{5.30}$$

where $\{w_1, \ldots, w_{N_w}\}$ are the vertices of the polytopes contained in $\mathcal{W}$. Therefore, we have that the optimization problem (5.13) can be rewritten into a QP in a form of

$$\min_{\widetilde{F}_i,\widetilde{g}_i} \sum_{j=1}^{N_w} \|\mu(w_j) - (\widetilde{F}_i x + \widetilde{g}_i)\|_2^2 \tag{5.31a}$$

$$\text{s.t. } \widetilde{F}_i v_{i,\ell} + \widetilde{g}_i \in \mathcal{U},\ \forall v_{i,\ell} \in \text{vert}(\widetilde{\mathcal{R}}_i), \tag{5.31b}$$

$$A v_{i,\ell} + B(\widetilde{F}_i v_{i,\ell} + \widetilde{g}_i) \in \mathcal{C}_\infty,\ \forall v_{i,\ell} \in \text{vert}(\widetilde{\mathcal{R}}_i). \tag{5.31c}$$

Remark 5.4.7 *The main difference between the aforementioned approaches is that in* (5.31) *we have used vertex enumeration of all intersected polytopes as in* (5.30)*, while in* (5.28) *we have preserved the general formulation of Problem 5.2.1 and employed triangulation* (5.19) *with formula from Baldoni et al. (2010), in order to proceed the integral. From the implementation point of view, the vertex approximation technique 5.4.1 is computationally less expensive, yet less general (cf. Remark 5.4.6). Their comparison will be also provided in Section 5.7.3 through one illustrate example.*

5.4.2 Properties of Fittted Explicit MPC Control Law

In Section 5.3 we have suggested how to synthesize polytopic partition $\widetilde{\Omega}$, which can be exploit in optimization problem (5.28) that yields explicit controller $\widetilde{u}(x)$. Here we will investigate the main properties of such approximated optimizer, which are:

Complexity

The complexity reduction of the control law $\mu(x)$ was the property with the highest priority (as the title of Chapter 5 indicates). We have that the complexity of $\mu(x)$ is reduced from M to $\widetilde{M}$ regions, since $\widetilde{u}(x)$ was obtained from the optimization problem (5.28), where a new polytopic partition $\widetilde{\Omega}$, constructed e.g. as suggested in Section 5.3 or randomly selected with respect to (5.7), has been employed. However, the price that we have paid, in order to achieve this goal, can be expressed by the value of the objective function (5.28a), which reflects the suboptimlity of $\widetilde{u}(x)$ with respect to $\mu(x)$.

Continuity

The optimization problem (5.28) does not a-priori guarantee a continuity of the approximated optimizer $\widetilde{u}(x)$, what may cause some difficulties. For example if $\widetilde{u}(x)$ is obtained as in (5.28) and we are given a state measurement that lies in between two regions $x \in \widetilde{\mathcal{R}}_i \cap \widetilde{\mathcal{R}}_j$, where the optimizer is discontinuous. Thus for this state we will obtain two different function values $\widetilde{F}_i x + \widetilde{g}_i \neq \widetilde{F}_j x + \widetilde{g}_j$ that could lead the controlled system out of balance (by switching from one control input to another), hence to cause a great mechanical stress. As consequence of this problem, we are forced to use an additional control logic (see Remark 4.3.1), that would decide, which of these control inputs should be applied to the system. Therefore, in order to prevent the aforementioned complication, one can enforce the continuity of $\widetilde{u}(x)$ by adding the additional constrains $\widetilde{F}_i w_k + \widetilde{g}_i = \widetilde{F}_j w_k + \widetilde{g}_j$ to constraints in (5.28), where w_k is a vector of vertices of the $n-1$ dimensional intersection $\widetilde{\mathcal{R}}_i \cap \widetilde{\mathcal{R}}_j$, $\forall i, j = 1, \ldots, \widetilde{M}$. Note that, since the simple feedback $\widehat{\mu}$ is continuous, the choice $\widetilde{F}_i = \widehat{F}_i$, $\widetilde{g}_i = \widehat{g}_i$ is a feasible continuous solution in (5.28). Needless to say, sacrificing continuity allows for a greater reduction of the approximation error in (5.28a).

Recursive satisfaction of original constraints

We have that $\widetilde{u}(x)$ provides recursive satisfaction of the original constraints in (5.1) due to (5.28b) and (5.28c), with respect to Lemma 5.4.3 and Definition 5.3.1.

Closed-loop stability

Here we need to distinguish two types of closed-loop stability, because such approximated optimizer $\widetilde{u}(x)$ provides bounded outputs for bounded inputs (BIBO), but, on the other hand, we do not have guarantee of asymptotic stability even though the original one $\widehat{\mu}(x)$, as in (5.8), did provide it. The asymptotic stability of $\widetilde{u}(x)$ can be provided e.g. by a posteriori certification that can be carried out by constructing a Lyapunov function to $\widetilde{u}(x)$, or by enforcing this property implicitly via adding constraints into the optimization problem (5.28).

5.5 Closed-Loop Stability

In Sections 5.3 and 5.4 we have shown the procedure, which yields a simpler, memory less demanding, control law $\widetilde{u}(x)$ that provides recursive satisfaction of the original state and input constraints, and furthermore guarantee that the approximation error, the suboptimality respectively, is minimized. Thus we have that $\widetilde{u}(x)$ satisfies criterion R1 and R3 of Problem 5.2.1 so far. On the other hand, as it was point out in Section 5.4.2, we have no a-priori guarantee of asymptotic closed-loop stability. Therefore, in this section we will show our further results (published in Holaza et al. (2015)), where the search for aforementioned simpler feedback law is enhanced such that R2 of Problem 5.2.1 holds.

To achieve such a property, we will assume that for system (5.11b) we have knowledge of a convex piecewise linear (PWL) Lyapunov function $V : \mathbb{R}^n \to \mathbb{R}$ with $\mathrm{dom}(V) \supseteq \mathcal{C}_\infty$. Such a PWL Lyapunov function can be straightforwardly obtained by considering the Minkowski function (also called the Gauge function) of $\mathcal{C}_\infty$ in (5.9). Let the minimal half-space representation of $\mathcal{C}_\infty$ be normalized to

$$\mathcal{C}_\infty = \{x \in \mathbb{R}^n \mid Wx \leq \mathbf{1}\}, \tag{5.32}$$

where $\mathbf{1}$ is a column vector of ones of appropriate dimension. Then $V(\cdot)$ is given, see Blanchini and Miani (2008), as

$$V(x) := \max_{k=1,\dots,d} w_k^T x, \tag{5.33}$$

where w_j^T denotes the j-th row of $W \in \mathbb{R}^{d\times n}$ in (5.32). It follows from Blanchini and Miani (2008); Lazar et al. (2008) that $V(\cdot)$ of (5.33) is a Lyapunov function for system (5.11b), with domain $\mathcal{C}_\infty$. Importantly, note that the number K of affine functions defining (5.33) is equal to d, the number of facets of $\mathcal{C}_\infty$.

Then it is well-known (see e.g. Lazar et al. (2008)) that $\widetilde{u}(x)$ will render the closed-loop system (5.4) asymptotically stable if

$$V(Ax + B\widetilde{u}(x)) \leq \gamma V(x) \tag{5.34}$$

holds for all $x \in \mathcal{C}_\infty$ and for some $\gamma \in [0, 1)$. By adding (5.34) to the constraints of (5.13), we can formulate the search for parameters $\widetilde{F}_i$, $\widetilde{g}_i$ of a stabilizing feedback $\widetilde{u}(x)$ in (5.3) as

$$\min_{\widetilde{F}_i,\widetilde{g}_i} \int_{\widetilde{\mathcal{R}}_i} \|\mu(x) - \widetilde{u}(x)\|_2^2 \,\mathrm{d}x \tag{5.35a}$$

$$\text{s.t. } \widetilde{F}_i x + \widetilde{g}_i \in \mathcal{U},\ \forall x \in \widetilde{\mathcal{R}}_i, \tag{5.35b}$$

$$Ax + B(\widetilde{F}_i x + \widetilde{g}_i) \in \mathcal{C}_\infty,\ \forall x \in \widetilde{\mathcal{R}}_i, \tag{5.35c}$$

$$V(Ax + B(\widetilde{F}_i x + \widetilde{g}_i)) \leq \gamma V(x),\ \forall x \in \widetilde{\mathcal{R}}_i, \tag{5.35d}$$

which needs to be solved for all regions $\widetilde{\mathcal{R}}_i$ of $\widetilde{u}(x)$.

Remark 5.5.1 *The constraint $V(Ax + B(\widetilde{F}_i x + \widetilde{g}_i)) \leq \gamma V(x)$ with $\gamma \in [0, 1)$ and $V(\cdot)$ as in (5.33) implicitly guarantees that $u = \widetilde{F}_i x + \widetilde{g}_i = 0$ for $x = 0$.*

Lemma 5.5.2 *With $\mathcal{U} = \{u \mid H_u u \leq h_u\}$ and $\mathcal{C}_\infty = \{x \mid Wx \leq \mathbf{1}\}$ as in (5.32), problem (5.35) is equivalent to*

$$\min_{\widetilde{F}_i,\widetilde{g}_i} \sum_{j\in\mathcal{J}_i} \sum_{k=1}^{K_{i,j}} \int_{\Delta_{i,j,k}} \|(F_j x - g_j) - (\widetilde{F}_i x + \widetilde{g}_i)\|_2^2 \,dx, \tag{5.36a}$$

$$s.t.\ H_u(\widetilde{F}_i v_{i,\ell} + \widetilde{g}_i) \leq h_u,\ \forall v_{i,\ell} \in \mathrm{vert}(\widetilde{\mathcal{R}}_i), \tag{5.36b}$$

$$W(Av_{i,\ell} + B(\widetilde{F}_i v_{i,\ell} + \widetilde{g}_i)) \leq \mathbf{1},\ \forall v_{i,\ell} \in \mathrm{vert}(\widetilde{\mathcal{R}}_i), \tag{5.36c}$$

$$w_k^T(Av_{i,\ell} + B(\widetilde{F}_i v_{i,\ell} + \widetilde{g}_i)) \leq \gamma \overline{m}_i,\ \forall k = 1, \ldots, d,\ \forall v_{i,\ell} \in \mathrm{vert}(\widetilde{\mathcal{R}}_i), \tag{5.36d}$$

with

$$\overline{m}_i = \max_{\ell=1,\ldots,n_{v,i}} \max_{k=1,\ldots,d} w_k^T v_{i,\ell}, \tag{5.37}$$

and $v_{i,1}, \ldots, v_{i,n_{v,i}}$ being the vertices of polytope $\widetilde{\mathcal{R}}_i$.

Proof. The first two constraints of (5.36) are equivalent to (5.28b) and (5.28c), respectively, and the objective function is the same as in (5.28a). Therefore it suffices to show that (5.36d) is equivalent to (5.35d). With $V(\cdot)$ as in (5.33), the constraint (5.35d) yields

$$\max_{k=1,\dots,d} w_k^T(Ax + B(\widetilde{F}_i x + \widetilde{g}_i)) \leq \gamma \max_{k=1,\dots,d} w_k^T x,\ \forall x \in \widetilde{\mathcal{R}}_i. \tag{5.38}$$

Since $\widetilde{\mathcal{R}}_i$ is assumed to be a polytope, and because the arguments of the maximum in the right-hand-side of (5.38) are linear functions of x, the maximum is attained at one of the vertices $\{v_{i,1}, \dots, v_{i,n_{v,i}}\}$ of $\widetilde{\mathcal{R}}_i$, and is denoted by $\overline{m}_i$ as in (5.37). Then, we can equivalently write (5.38) as

$$\max_{k=1,\dots,d} w_k^T(Ax + B(\widetilde{F}_i x + \widetilde{g}_i)) \leq \gamma \overline{m}_i,\ \forall x \in \widetilde{\mathcal{R}}_i. \tag{5.39}$$

Next, denote $f(x) := \max_{k=1,\dots,d} w_k^T(Ax + B(\widetilde{F}_i x + \widetilde{g}_i))$ and recall that the maximum of affine functions is a convex function (Boyd and Vandenberghe, 2004, Section 3.2.3). With $f(\cdot)$ convex, it is trivial to show that $f(x) \leq \overline{m}_i$ for all $x \in \widetilde{\mathcal{R}}_i$ if and only if $f(v_{i,\ell}) \leq \overline{m}_i$ for all vertices $v_{i,1}, \dots, v_{i,n_{v,i}}$ of polytope $\widetilde{\mathcal{R}}_i$. Hence (5.39) is equivalent to

$$\max_{k=1,\dots,d} w_k^T(Av_{i,\ell} + B(\widetilde{F}_i v_{i,\ell} + \widetilde{g}_i)) \leq \gamma \overline{m}_i,\ \forall v_{i,\ell} \in \mathrm{vert}(\widetilde{\mathcal{R}}_i). \tag{5.40}$$

Finally, since $\max_k w_k^T z \leq \overline{m}_i$ holds if and only if $w_k^T z \leq \overline{m}_i$ is satisfied for all k, we get

$$w_k^T(Av_{i,\ell} + B(\widetilde{F}_i v_{i,\ell} + \widetilde{g}_i)) \leq \gamma \overline{m}_i,\ \forall k = 1, \dots, d,\ \forall v_{i,\ell} \in \mathrm{vert}(\widetilde{\mathcal{R}}_i), \tag{5.41}$$

which is precisely the same as in (5.36d).

Remark 5.5.3 *Note that $\overline{m}_i$ in* (5.37) *can be computed analytically once the vertices of $\widetilde{\mathcal{R}}_i$ are known.*

For each region $\widetilde{\mathcal{R}}_i$, (5.36) is a QP for the objective function (5.36a) is quadratic and all constraints in (5.36) are linear functions of $\widetilde{F}_i$ and $\widetilde{g}_i$. Hence the search for parameters $\widetilde{F}_i$, $\widetilde{g}_i$ of a stabilizing simpler feedback $\widetilde{u}(x)$ of (5.3) can be formulated as a series of $\widetilde{M}$ quadratic programs, as captured by the following theorem.

Theorem 5.5.4 *Suppose that the quadratic programs in* (5.36) *are feasible for all regions $\widetilde{\mathcal{R}}_i$, $i = 1, \dots, \widetilde{M}$ and for a selected $\gamma \in [0, 1)$. Then the refined simpler*

feedback $\widetilde{u}(x)$ *of* (5.3) *provides recursive satisfaction of state and input constraints in* (5.1d) *and* (5.1c)*, attains asymptotic stability of the closed-loop system in* (5.4)*, and minimizes the integrated squared error in* (5.5)*.*

Proof. The first two constraints in (5.36) are the same as in (5.28) and enforce recursive feasibility according to Theorem 5.4.4. Similarly, minimization of the integrated squared error is the same as in (5.36a). Finally, feasibility of (5.36) implies that there exist parameters $\widetilde{F}_i$, $\widetilde{g}_i$ of $\widetilde{u}(x)$ which enforces a given decay of the Lyapunov function by (5.34) and by Lemma 5.5.2.

Unlike Theorem 5.4.4 which provides necessary and sufficient conditions, feasibility of QPs (5.36) is merely sufficient for existence of $\widetilde{u}(\cdot)$ that renders the closed-loop system asymptotically stable. If the QPs are infeasible, one can enlarge the value of γ, provided it fulfills $\gamma \in [0, 1)$, or alternatively employ a new partition $\{\widetilde{\mathcal{R}}_i\}_i^{\widetilde{M}}$ obtained for a different value of $\widehat{N}$ in Lemma 5.3.3.

5.6 Complete Procedure

Here we can summarize the complete procedure, which we have defined in Section 5.3, Section 5.4 and Section 5.5 respectively. Therefore, if one is interested in a simpler control law $\widetilde{u}(x)$ that satisfies all criterions stated in Problem 5.2.1, we suggest to proceed the Algorithm 7.

We note that complex optimizer is defined over polytopic partition Ω, which is composed of M regions $\mathcal{R}_j$, thus $j = 1, \ldots, M$. Furthermore in steps 5 and in 7-10, (11-14 respectively) of the Algorithm 7, we consider only the non-empty intersection $\mathcal{Q}_{i,j}$, which are denoted by $\mathcal{J}_i$ as in (5.15). Obtaining the polytopes $\widetilde{\mathcal{R}}_i$ in step 1 by solving (5.1) explicitly can be performed e.g. by the MPT Toolbox (Herceg et al., 2013a) or by the Hybrid Toolbox (Bemporad, 2003). Computation of intersections, tessellation (via Delaunay triangulation), and enumeration of vertices in steps 4 and 5 can also be done by MPT. Finally, the optimization problem (5.28) can be formulated by YALMIP (Löfberg, 2004) and solved using off-the-shelf software, e.g., by GUROBI (Gurobi Optimization, 2014) or `quadprog` of MATLAB.

Recalling that the feasibility of Algorithm 7 depends on the selection of polytopic partition $\widetilde{\Omega}$ and the stability parameter γ. Moreover, we need to keep in mind that by varying of the parameter γ or the partition $\widetilde{\Omega}$, we can adjust the performance of $\widetilde{u}(x)$ as well.

Algorithm 7: Complete procedure of complexity reduction.

Input : Complex optimizer $\mu(x)$

Output: Derived optimizer $\widetilde{u}(x)$

Solve (5.1) with $\hat{N} < N$ to obtain $\widetilde{\Omega}$ composed of $\widetilde{M}$ regions $\widetilde{\mathcal{R}}_i$;
Select $\gamma \in [0, 1)$ (if closed-loop stability is desired);
for $i = 1, \ldots, \widetilde{M}$ **do**
    Compute $\mathcal{Q}_{i,j}$ from (5.14) for each $j = 1, \ldots, M$;
    Triangulate each intersection $\mathcal{Q}_{i,j}$ into n-simplices $\Delta_{i,j,1}, \ldots, \Delta_{i,j,K}$ and enumerate their respective vertices $s_1, \ldots, s_{n+1}$;
    Enumerate vertices $\mathcal{V}_i = \{v_{i,1}, \ldots, v_{i,n_{v,i}}\}$ of $\widetilde{\mathcal{R}}_i$;
    if *only recursive feasibility is desired* **then**
        Obtain the analytic expression of the integrals in (5.28a);
        Solve QP (5.28) to obtain parameters $\widetilde{F}_i$ and $\widetilde{g}_i$;
    end
    if *closed-loop stability is desired* **then**
        Obtain the analytic expression of the integrals in (5.36a) by (5.17);
        Solve QP (5.36) to obtain parameters $\widetilde{F}_i$ and $\widetilde{g}_i$;
    end
end
return $\widetilde{u}(x) = \widetilde{F}_i x + \widetilde{g}_i, \;\; i = 1, \ldots, \widetilde{M}$;

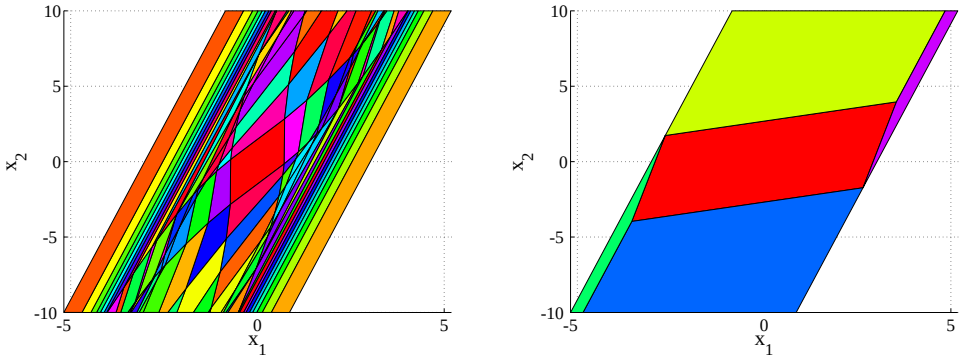
5.7 Case Studies

In this section we demonstrate effectiveness of the presented explicit MPC complexity reduction method on two examples with different number of states. Furthermore, we provide an additional example, where two different QP formulations of (5.28) and (5.31) are compared.

5.7.1 Two-Dimensional Example

Consider the second order, discrete-time, linear time-invariant system

$$x(t+1) = \begin{bmatrix} 0.9539 & -0.3440 \\ -0.4833 & -0.5325 \end{bmatrix} x(t) + \begin{bmatrix} -0.4817 \\ -0.5918 \end{bmatrix} u(t), \tag{5.42}$$



(a) 127 regions of $\mu(x)$. (b) 5 regions of $\widetilde{u}(x)$.

Figure 5.2: Regions of the complex controller $\mu(x)$ and of the approximate feedback $\widetilde{u}(x)$.

which is subject to state constraints $-10 \leq x_i \leq 10$, $i = 1, 2$ and input bounds $-0.5 \leq u \leq 0.5$. We remark that the system is open-loop unstable with eigenvalues $\lambda_1 = 1.0584$ and $\lambda_2 = -0.6370$. The complex explicit MPC controller $\mu(x)$ in (5.2) was obtained by solving (5.11) for $Q_x = I_2$, $Q_u = 2$ and $N = 20$. Its explicit representation was defined over $M = 127$ polytopic regions $\mathcal{R}_i \subset \mathbb{R}^2$, shown in Figure 5.2(a). All computations were carried out on a 2.7 GHz CPU using MATLAB and the MPT Toolbox.

To derive a simple representation of the MPC feedback as in (5.3), we have proceeded as outlined in Section 5.6. First, we have solved (5.11) with shorter prediction horizons $\hat{N} \in \{1, 2, 3, 4\}$. This provided us with simple feedbacks $\widehat{\mu}(x)$ as in (5.8) with lower performances. The domains of these feedbacks were defined, respectively, by $\widetilde{M} = \{3, 5, 11, 17\}$ regions $\widetilde{\mathcal{R}}_i$. These regions were then employed in (5.36) to optimize the parameters $\widetilde{F}_i$, $\widetilde{g}_i$ of the improved simple feedbacks $\widetilde{u}(x)$ in (5.3) while guaranteeing closed-loop stability. The fitting problems (5.36) were formulated by YALMIP and solved by `quadprog`.

Remark 5.7.1 *In practice, to get the least complex approximate controller $\widetilde{u}(x)$ one would only consider the case with the smallest number of regions. We only consider various values of $\widetilde{M}$ to assess the suboptimality of $\widetilde{u}(x)$ with respect to $\mu(x)$ as a function of the number of regions, $\widetilde{M}$.*

Next, we have assessed the degradation of performance induced by employing

Pred. horizon	# of regions	Suboptimality w.r.t. $\mu(x)$ in (5.2)	
		$\widehat{\mu}(x)$ from Lemma 5.3.3	$\widetilde{u}(x)$ from (5.36)
1	3	60.8 %	25.1 %
2	5	32.9 %	18.0 %
3	11	11.4 %	8.3 %
4	17	6.9 %	1.7 %

Table 5.1: Complexity and suboptimality comparison for the example in Section 5.7.1. The (complex) optimal controller consisted of 127 regions.

simpler feedbacks $\widehat{\mu}(x)$ and $\widetilde{u}(x)$ instead of the optimal controller $\mu(x)$. To do so, for each suboptimal controller we have performed closed-loop simulations for 10000 equidistantly spaced initial conditions from the domain of $\mu(x)$. In each simulation we have evaluated the performance criterion $J_{\text{sim}} = \sum_{i=1}^{N_{\text{sim}}} x_i^T Q_x x_i + u_i^T Q_u u_i$ for $N_{\text{sim}} = 100$. For each investigated controller we have subsequently computed mean values of this criterion over all investigated starting points. This "average" performance indicators are denoted in the sequel as J_{opt} for the optimal feedback $\mu(x)$, J_{simple} for the simple, but suboptimal controller $\widehat{\mu}(x)$, and J_{improved} for $\widetilde{u}(x)$, whose parameters were optimized in (5.36). Then we can express the average suboptimality of $\widetilde{u}(x)$ by $J_{\text{simple}}/J_{\text{opt}}$, and the suboptimality of $\widetilde{u}(x)$ by $J_{\text{improved}}/J_{\text{opt}}$. The higher the value, the larger the suboptimality of the corresponding controller is with respect to the optimal feedback $\mu(x)$.

Concrete numbers are reported in Table 5.1. As can be observed, lowering the prediction horizon significantly reduces complexity. However suboptimality is inverse-proportional to complexity. For instance, solving (5.11) with $N = 1$ gives $\widehat{\mu}(x)$ that performs by 60 % worse compared to the optimal feedback $\mu(x)$ obtained for $N = 20$. Improving parameters of the feedback function via (5.36) resulted in an improved controller $\widetilde{u}(x)$ whose average suboptimality is only 25 %. The amount of suboptimality can be further reduced by considering more complex partition of the feedback function. In all cases reported in Table 5.1 the simpler feedback $\widetilde{u}(x)$ with mitigated suboptimality guarantees closed-loop stability since the corresponding fitting problems (5.36) were feasible for $\gamma < 1$.

5.7.2 Inverted Pendulum on a Cart

Next we consider an inverted pendulum mounted on a moving cart, shown in Figure 5.3. Linearizing the nonlinear dynamics around the upright, marginally stable position leads to the following linear model:

$$\begin{bmatrix} \dot{p}(t) \\ \ddot{p}(t) \\ \dot{\phi}(t) \\ \ddot{\phi}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -0.182 & 2.673 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -0.455 & 31.182 & 0 \end{bmatrix} \begin{bmatrix} p(t) \\ \dot{p}(t) \\ \phi(t) \\ \dot{\phi}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1.818 \\ 0 \\ 4.546 \end{bmatrix} u(t), \tag{5.43}$$

where $p(t)$ is the position of the cart, $\dot{p}(t)$ is the cart's velocity, $\phi(t)$ is the pendulum's angle from the upright position, and $\dot{\phi}(t)$ denotes the angular velocity. The control input $u(t)$ is proportional to the force applied to the cart. System (5.43) is then discretized by assuming sampling time 0.1 seconds.

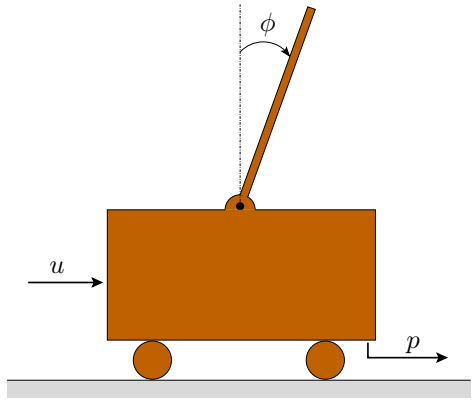


Figure 5.3: Inverted pendulum on a cart.

The optimal (complex) controller $\mu(x)$ in (5.2) was then constructed by solving (5.11) with prediction horizon $N = 8$, penalties $Q_x = \mathrm{diag}(10, 1, 10, 1)$, $Q_u = 0.1$, and constraints $|p(t)| \leq 1$, $|\dot{p}(t)| \leq 1.5$, $|\phi|(t) \leq 0.35$, $|\dot{\phi}(t)| \leq 1$, $|u(t)| \leq 1$. Using the MPT toolbox we have obtained $\mu(x)$ defined over 943 polytopes of the 4-dimensional state-space.

Subsequently we have constructed simple feedback controllers $\widehat{\mu}(x)$ according to Lemma 5.3.3 for prediction horizons $\hat{N} \in \{1, 2, 3\}$. This provided us with polytopic partitions $\{\widetilde{\mathcal{R}}_i\}$ defined, respectively, by 35 polytopes for $\hat{N} = 1$, 117 regions $\hat{N} = 2$, and 273 polytopes in case of $\hat{N} = 3$. For each partition we have then optimized

Prediction horizon	Number of regions	$\widehat{\mu}(x)$ from Lemma 5.3.3		$\widetilde{u}(x)$ from (5.36)	
		AST	Suboptimality	AST	Suboptimality
1	35	8.3 s	159.4 %	5.1 s	59.4 %
2	117	4.6 s	43.8 %	3.7 s	15.6 %
3	273	3.5 s	9.4 %	3.4 s	6.3 %

Table 5.2: Complexity and suboptimality comparison for the example in Section 5.7.2. The (complex) optimal controller consisted of 943 regions.

the gains $\widetilde{F}_i$, $\widetilde{g}_i$ of $\widetilde{u}(x)$ in (5.3) by solving (5.28).

To assess the degradation of performance induced by employing the simpler controllers instead of the optimal feedback, we have performed 100 closed-loop simulations for various values of the initial cart's position p. Then we have measured the number of simulation steps in which a particular controller drives all states into the ± 0.01 neighborhood of the origin. In other words, our performance-evaluation criterion measures liveness properties of a particular controller. The average settling time for the optimal (complex) feedback $\mu(x)$ was 32 sampling times (which corresponds to 3.2 seconds). The aggregated results showing performance of the two simple feedbacks $\widehat{\mu}(x)$ and $\widetilde{u}(x)$ are reported in Table 5.2. The columns of the table represent, respectively, the prediction horizon $\hat{N}$, number of polytopes over which both simple controllers are defined, as well as performance of the simple feedback $\widehat{\mu}(x)$ in (5.8). Here, AST stands for Average Settling Time, and the suboptimality percentage represents the relative increase of the settling time compared to AST = 3.2 seconds for the optimal (complex) feedback. The final two columns show performance of $\widetilde{u}(x)$, whose gains were optimized by (5.28). As it can be seen, refining the gains $\widetilde{F}_i$, $\widetilde{g}_i$ via (5.28) significantly mitigates the degradation of performance.

To illustrate the differences in performance of the three controllers, Figure 5.4 shows closed-loop profiles of states and inputs under $\mu(x)$, $\widehat{\mu}(x)$, and $\widetilde{u}(x)$ for the initial conditions $p(0) = 0.525$, $\dot{p}(0) = 0$, $\phi(0) = 0$, $\dot{\phi}(0) = 0$. Here, we have employed the second case of Table 5.2 where $\widehat{\mu}(x)$ and $\widetilde{u}(x)$ were both defined over 117 polytopes. Comparing the state profiles in Figures 5.4(a), 5.4(c), and 5.4(e) we can clearly see the benefit of refining the gains of $\widetilde{u}(x)$ via (5.28). In particular, the performance of $\widetilde{u}(x)$ derived according to Section 5.4 is nearly identical to the

performance of the optimal (complex) feedback $\mu(x)$. The simple feedback $\widehat{\mu}(x)$, on the other hand, performs significantly worse. We remind that in all cases shown in Table 5.2, the complexity of $\widetilde{u}(x)$ is significantly smaller than the number of regions of the optimal feedback (which was defined over 943 polytopes).

5.7.3 Comparation Example

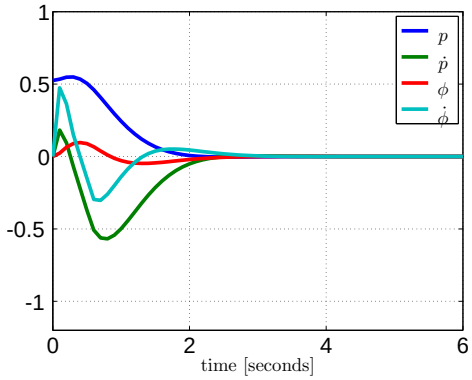
In this section we show the comparision of two different approaches of the criterion (5.5) in Problem 5.2.1. The first one (5.28) exploids the integral as in (5.17), while the other one (5.4.1) applies only the sum of squares (5.29) in all intersected vertices $\mathcal{W}$, in order to minimize the suboptimality of such derived optimizer $\widetilde{u}$. For this purpose we have employed the oscillating system, with the same setup as in Holaza et al. (2013), that is depicted in Figure 5.5. Thus, we have the second order, discrete-time, linear time-invariant system

$$x(t+1) = \begin{bmatrix} 0.5403 & -0.8415 \\ 0.8415 & 0.5403 \end{bmatrix} x(t) + \begin{bmatrix} -0.4597 \\ 0.8415 \end{bmatrix} u(t), \tag{5.44}$$

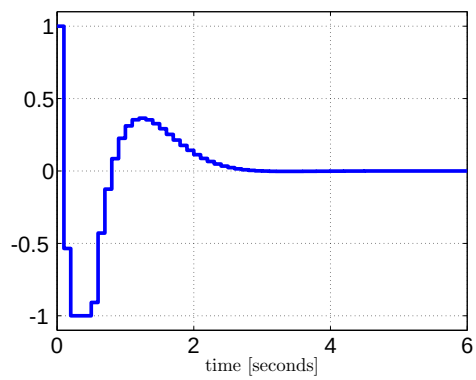
where for the first state we require $-10 \leq x_1 \leq 10$, and the input is bounded by $-1 \leq u \leq 1$. The complex optimizer $\mu(x)$ in (5.2) was obtained by solving (5.11) for $Q_x = I_2$, $Q_u = 1$ and $N = 10$. Its explicit representation was defined over $M = 159$ polytopic regions $\mathcal{R}_i \subset \mathbb{R}^2$ as shows Figure 5.6(a).

Since the procedure of deriving the simpler polytopic partition in 5.3 is the same for both approaches, we proceeded as it is here specified and solved the same optimization problem (5.11) but with shorter prediction horizon $\hat{N} \in \{2, 3, 5, 7, 9\}$. This resulted in a new partitioning of the original domain into $\widetilde{M} = \{7, 13, 33, 77, 131\}$ regions (e.g. the polytopic partition composed of 7 regions is depicted in Figure 5.6(b)) over which the explicit feedback $\hat{u}(x)$ was defined. Note that for each choice of $\hat{N}$ we have $\widetilde{M} < M$. Subsequently, for each of these partitions we have optimized parameters $\widetilde{F}_i$, $\widetilde{g}_i$ of $\widetilde{u}(x)$ in (5.2) via solving the fitting problem (5.28) and (5.31) respectively. Since, two different approach are used to obtain $\widetilde{u}(x)$, denote $\widetilde{u}_I(x)$ as the solution of problem (5.28), while $\widetilde{u}_V(x)$ the solution of problem (5.31).

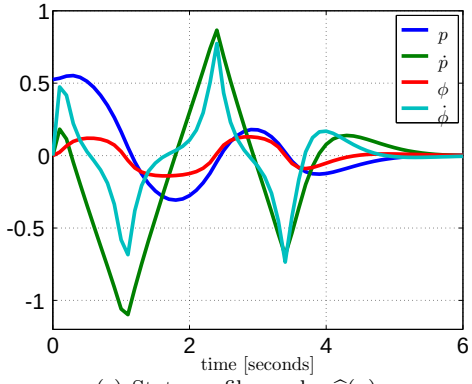
The performance degradation is here also induced by means of substituting simpler optimizers, instead of $\mu(x)$, into (5.44) and performing closed-loop simulations over $N_{\text{sim}} = 100$ simulation steps for 10000 equidistantly spaced feasible initial conditions $x(0) \in \text{dom}(\widetilde{u})$. All respective closed-loop profiles of states



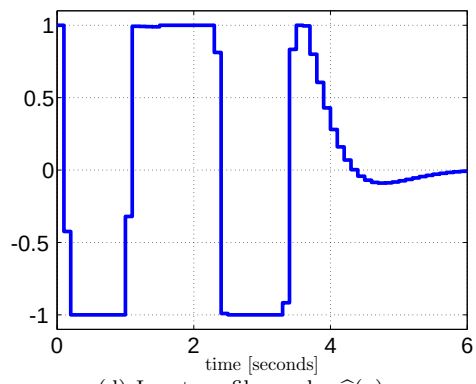
(a) State profiles under $\mu(x)$.



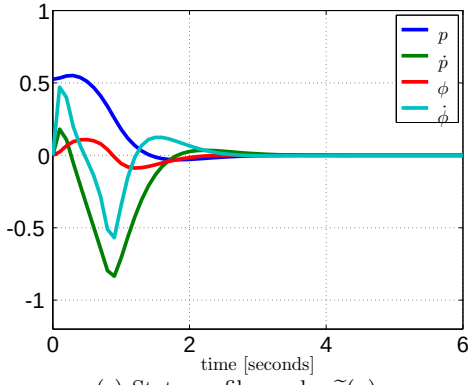
(b) Input profiles under $\mu(x)$.



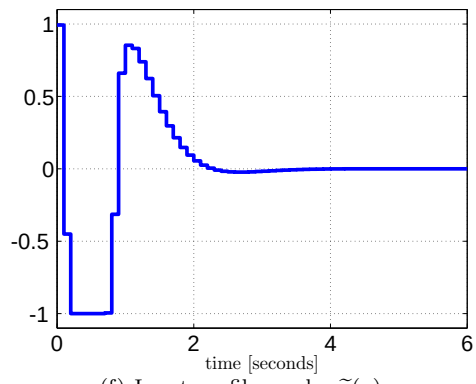
(c) State profiles under $\widehat{\mu}(x)$.



(d) Input profiles under $\widehat{\mu}(x)$.



(e) State profiles under $\widetilde{u}(x)$.



(f) Input profiles under $\widetilde{u}(x)$.

Figure 5.4: Simulated closed-loop profiles of pendulum's states and inputs under the (complex) optimal feedback $\mu(x)$, under the simple controller $\widehat{\mu}(x)$ in (5.8), and under its optimized version $\widetilde{u}(x)$ obtained from (5.28).

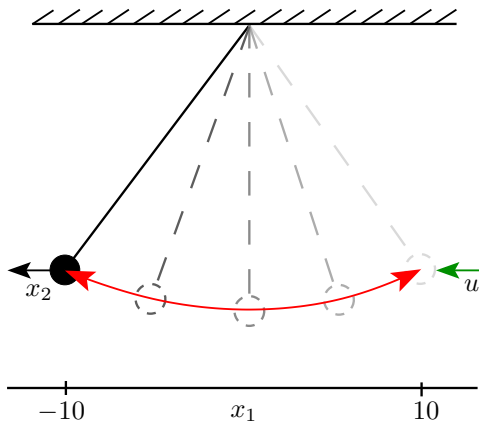
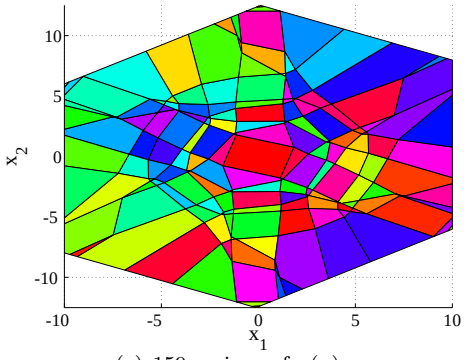
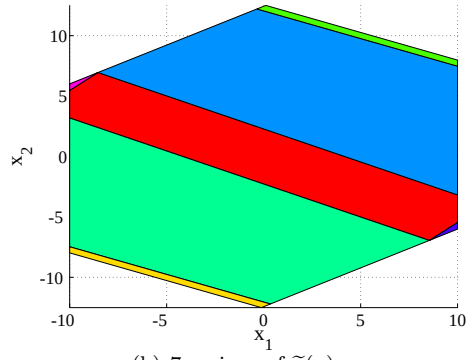


Figure 5.5: Illustration of a pendulum system.



(a) 159 regions of $\mu(x)$.



(b) 7 regions of $\widetilde{u}(x)$.

Figure 5.6: Regions of the complex optimizer and of the least complex approximate optimizer.

Pred. horizon	# of regions	Suboptimality w.r.t. $\mu(x)$ in (5.2)		
		$\widetilde{u}_I(x)$ from (5.31)	$\widetilde{u}_V(x)$ from (5.28)	$\Delta\widetilde{u}(x)$
1	3	85.83 %	85.71 %	0.12 %
2	7	0.39 %	0.39 %	0.00 %
3	13	0.19 %	0.30 %	−0.11 %
4	23	0.19 %	0.34 %	−0.15 %
5	33	0.09 %	0.19 %	−0.10 %
6	53	0.04 %	0.09 %	−0.05 %
7	77	0.03 %	0.09 %	−0.06 %

Table 5.3: Complexity and suboptimality comparison for the example in Section 5.7.3. The (complex) optimal controller consisted of 159 regions. Here the degree of the difference is expressed by $\Delta\widetilde{u}(x) = \widetilde{u}_I(x) - \widetilde{u}_V(x)$.

and control inputs were then employed to access the performance criterion $J_{\text{sim}} = \sum_{i=1}^{N_{\text{sim}}} x_i^T Q_x x_i + u_i^T Q_u u_i$. By computing the mean values of J_{sim} for all investigated initial conditions, we have obtained the overall (average) performance of J_{opt} for the optimal feedback $\mu(x)$, J_{simple} for $\widehat{\mu}(x)$, $J_{\text{improved},I}$ for $\widetilde{u}_I(x)$, and $J_{\text{improved},V}$ for $\widetilde{u}_V(x)$.

The results are reported in Table 5.3. Here we have proven the efficacy of the proposed method once again, since the complexity of both approximated optimizers are significantly reduced while the suboptimality is still preserved at a reasonable level. The Table 5.3 however mainly shows the difference of two distinguish approximation formulations. As can be seen the rendered optimizer $\widetilde{u}_I(x)$ emerged as a winner from the given comparison, since in all scenarios (except of one with $N = 1$) has shown better performance in respect to in respect to $\widetilde{u}_V(x)$. This fact is emphasized in the Table 5.3 by a negative value of $\Delta\widetilde{u}(x)$. Thus, we have that the employed integral formulation (5.31) and (5.36) respectively are more suitable when one wants to reduce the complexity of a given complex explicit optimizer $\mu(x)$.

5.8 Conclusions

In this chapter, we have addressed the problem of reducing complexity of explicit MPC controllers. Particularly, we have sugested a novel memory reduction technique that is based on replacing the complex explicit feedback law $\mu(x)$ by its simpler, yet suboptimal, representation $\widetilde{u}(x)$. We have proposed to approach the given problem via construction a new polytopic partition $\widetilde{\Omega}$ such that (5.7) was satisfied. To achieve this, the same optimization problem (5.11), which was used to obtain complex optimizer $\mu(x)$, was solved once again with a shorter prediction horizon $\hat{N}$. Even though that we have already obtain a simpler control feedback $\widehat{\mu}(x)$, which provided recursive satisfaction of original state and input constraints, we were forced to discard it as it exhibited a great performance deterioration, compared to $\mu(x)$. This problem was addressed in Section 5.4 and Section 5.4.1, respectively, where we have suggested how one can significantly reduce the amount of suboptimality. Here we have shown, that the search for new local affine expressions of $\widetilde{u}(x)$ can be formulated as a quadratic optimization problem that entails conditions of recursive feasibility and closed-loop stability, as proposed in Section 5.5.

The complete procedure, was summarized in Section 5.6. Here, it was discussed that Algorithm 7 is always feasible if maximal control invariant set is employed in (5.11d). On the other hand, if closed-loop stability was required, then the feasibility depends on the selection of parameter $\gamma \in [0, 1)$.

Finally, the efficiency of the proposed technique has been demonstrated on three illustrative examples. Here we have chosen several prediction horizons $\hat{N}$ in order to vary the polytopic partition $\widetilde{\Omega}$, hence to obtain different $\widetilde{u}(x)$. All results have been saved in Tables 5.1, 5.2 and 5.3, where one can easily observe the potential of this technique.

Chapter 6

Verification Techniques in Model Predictive Control

It is well known that MPC is one of the most advanced control strategies, which nowadays represents an accepted standard in the industries. Its popularity was raised from the natural capability of designing control feedbacks for the multivariable systems and embedding all of its constraints into the optimization problem. Yet, in practice, we might come into contact with systems which are tangled with non-trivial limitations or performance requirements, and in order to devise a suitable controller, one need to impose all of these specifications into the optimization problem. This, however, results in too complex mathematical formulation what makes MPC often prohibitive for application on simple industrial hardware which possesses only limited computational resources.

This obstacle can be overcome e.g. via convex relaxations or by simply neglecting certain constraints, the consequence of which however leads to controllers that do not posses a-priori guarantees all of desired properties, yet some of them can be evoked by e.g. a proper tuning (Section 3.3.2). Needless to say, such simplified control policies might represent a risk and danger not only for company profit or environment but also for a human life. It is, therefore, essential to firstly certify these control laws, whether they provide all necessary properties before they are applied to a process. This issue is referred as a *certification problem* and in this section we will enrich this field by two novel methods.

In the first proposed method we investigate how the performance of explicit MPC feedback laws is affected by rounding-based quantization. Specifically, we address the problem of providing a rigorous certificate that a given piecewise affine explicit MPC feedback is bounded from below and from above by specific functions. These functions are constructed as to reflect typical control requirements, such as recursive feasibility, closed-loop stability, or bounded deterioration of performance. We show how to obtain an analytical form of the quantized MPC feedback and how to provide the certificate by solving a set of linear programs.

In the second method we show for a closed-loop system composed of a linear controlled plant and an MPC feedback strategy how to verify that closed-loop state trajectories either enter or avoid a given set of unsafe states. The search for the safety certificate is formulated as a mixed-integer linear programming problem which yields non-conservative certificates. The optimal control commands generated by the MPC policy are represented by Karush-Kuhn-Tucker optimality conditions, which allow to perform the verification without the need to explicitly compute reachable sets.

6.1 Verification of A-Posteriori Quantized Explicit MPC Feedback Laws

Control under quantized feedbacks is an important research field since a majority of control policies is nowadays implemented on digital platforms, which inherently induce quantization effects due to finite-precision arithmetics and employment of analog-to-digital and digital-to-analog conversions. Numerous techniques for designing quantized feedback strategies which provide desired properties (e.g., closed-loop stability and/or performance) were proposed recently, see e.g. (Goodwin and Quevedo, 2003; Nair et al., 2007) and references therein. Typically, the approaches consider the control of linear time-invariant systems by a linear feedback, quantized by memory-less static quantizers. When such a setup is considered, (Huijun and Tongwen, 2008) have shown how to derive maximum sectors bounds for which the quantized linear feedback attains closed-loop stability. In (Fu and Xie, 2009) the authors have studied which number of quantization levels is required to attain stability. If the feedback law is PWA, and the system to be controlled contains just one state and one input, (Fagnani and Zampieri, 2003) have developed conditions

under which the quantized feedback attains practical stability in the sense of providing guarantees that the closed-loop trajectories converge to certain intervals. A common deficiency of the aforementioned approaches, though, is that they do not consider constraints on system's states and inputs.

On the other hand, explicit MPC excels at providing optimal performance while rigorously enforcing constraint satisfaction (cf. Chapter 4). The limitation of traditional explicit MPC techniques for quantized systems is in the inherent complexity of the off-line optimization (Kvasnica, 2009). In particular, the MPC piecewise affine feedback needs to be obtained by solving a parametric mixed-integer optimization problem, where the individual quantization levels are modeled by binary (or integer) variables. The total number of such discrete components is then proportional to the prediction horizon. As the horizon or the number of states increase, the problems become very challenging to solve. One notable exception is the work of (Quevedo et al., 2002), in which the explicit representation of the quantized feedback is developed directly by employing Voronoi diagrams. The limitation, however, is that the resulting quantized control law does not posses a-priori guarantees of recursive feasibility and closed-loop stability.

Instead of attempting to devise a quantized explicit MPC feedback directly, in this method we propose to employ the real-valued MPC feedback, synthesized for a linear system subject to real-valued control inputs. The optimal real-valued feedback is then quantized, a-posteriori, by a static memory-less quantizer with a finite number of quantization levels. The main objective is to certify that such an a-posteriori quantized feedback achieves desired properties. If a positive certificate is obtained, one can safely apply the real-valued feedback, whose construction is much simpler compared to designing a quantized feedback directly. To provide a meaningful certificate, we first develop the analytical form of the a-posteriori quantized PWA feedback law in Section 6.1.2. Subsequently, in Section 11 we show how to certify whether the quantized PWA function is bounded from above and from below by given PWA functions by solving a finite number of linear programs. The certification procedure is non-conservative. Then, in Section 6.1.3 we illustrate how suitable bounding functions are constructed as to reflect requirements of recursive satisfaction of state and input constraints, or achievement of closed-loop stability and bounded deterioration of performance.

6.1.1 Problem Statement

We aim at controlling linear time-invariant systems in the discrete-time domain described by the state-update equation

$$x(t+1) = Ax(t) + Bu(t), \tag{6.1}$$

where $x(t) \in \mathbb{R}^n$ are the states and $u(t) \in \mathbb{R}^m$ are the control inputs, with $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, and the pair (A, B) controllable. The system in (6.1) is subject to constraints

$$x(t) \in \mathcal{X},\ u(t) \in \mathcal{U}, \tag{6.2}$$

where $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{U} \subseteq \mathbb{R}^m$ are non-empty polyhedra that contain the origin in their respective interiors.

For the system (6.1), the finite-horizon MPC problem can be formulated as

$$\min_{u_0,\dots,u_{N-1}} x_N^T Q_{\mathrm{N}} x_N + \sum_{k=0}^{N-1} x_k^T \left(Q_{\mathrm{x}} x_k + u_k^T Q_{\mathrm{u}} u_k\right) \tag{6.3a}$$

$$\text{s.t. } x_{k+1} = Ax_k + Bu_k,\ k = 0,\dots,N-1, \tag{6.3b}$$

$$x_k \in \mathcal{X},\ k = 0,\dots,N-1, \tag{6.3c}$$

$$u_k \in \mathcal{U},\ k = 0,\dots,N-1, \tag{6.3d}$$

$$x_N \in \mathcal{X}_f, \tag{6.3e}$$

where x_k and u_k denote, respectively, the state and input predictions at the k-th step. Q_{N} is the terminal penalty, Q_{x} and Q_{u} are stage penalties, and $\mathcal{X}_f \subset \mathbb{R}^n$ is the polytopic terminal set.

It is well known (see Section 4.1.4) that by applying multiparametric programming, the MPC feedback law[1]

$$u = \mu(x), \tag{6.4}$$

with $\mu : \mathbb{R}^n \to \mathbb{R}^m$ can be obtained by solving (6.3) as a mp-QP. Then $\mu(\cdot)$ is a PWA function of the state, given as

$$\mu(x) := \begin{cases} F_1 x + g_1 & \text{if } x \in \mathcal{R}_1, \\ \quad\vdots & \\ F_M x + g_M & \text{if } x \in \mathcal{R}_M, \end{cases} \tag{6.5}$$

[1] To simplify the notation, we will henceforth assume $u = u(t)$ and $x = x(t)$.

where $F_i \in \mathbb{R}^{m\times n}$, $g_i \in \mathbb{R}^m$, $\mathcal{R}_i \subseteq \mathbb{R}^n$, $i = 1, \ldots, M$, and the polyhedra $\mathcal{R}_i$ do not overlap.

Assume now that we are given a total of d quantization levels $q_1, \ldots, q_d$, $q_i \neq q_j$ $\forall i \neq j$. Then the rounding-based quantized version of (6.4) is

$$u = \mu_{\rm q}(x) := \begin{cases} q_1 & \text{if } x \in \mathcal{P}_i, \\ \quad \vdots & \\ q_d & \text{if } x \in \mathcal{P}_d, \end{cases} \tag{6.6}$$

where

$$\mathcal{P}_i = \{x \mid \|\mu(x) - q_i\|_2 \leq \|\mu(x) - q_j\|_2,\ \forall j \neq i\}, \tag{6.7}$$

denotes the region of the state space where the control command generated by the real-valued feedback (6.4) is closer to the i-th quantization level than to any other level. In other words, (6.6) rounds the value of (6.4) to the nearest quantization level. Note that each region $\mathcal{P}_i$ in (6.7) can, in general, be a non-convex and disconnected set. However, the regions can be decomposed into a finite number D of connected sets. If $\mu(\cdot)$ in (6.4) is a linear function, then $D = d$. Otherwise, $D \geq d$.

The objective of the proposed method is illustrated in Figure 6.1 and can be formally stated as follows:

Problem 6.1.1 *Given are: the real-valued explicit MPC feedback* $\mu(\cdot)$ *in* (6.4) *and* (6.5)*, performance bounds* $\underline{V} : \mathbb{R}^n \to \mathbb{R}^m$, $\overline{V} : \mathbb{R}^n \to \mathbb{R}^m$ *with* $\mathrm{dom}(\underline{V}) = \mathrm{dom}(\overline{V}) = \mathrm{dom}(\mu_q) = \Omega$*, and the quantization levels* $q_1, \ldots, q_d$*. Determine whether the quantized feedback* (6.6) *satisfies performance bounds*

$$\underline{V}(x) \leq \mu_q(x) \leq \overline{V}(x),\ \forall x \in \Omega. \tag{6.8}$$

In Section 6.1.3 we will show how to design the bounds $\underline{V}(\cdot)$, $\overline{V}(\cdot)$ as to capture various performance and safety criteria, including bounded deterioration of performance, satisfaction of constraints or closed-loop stability.

The difficulty of solving Problem 6.1.1 stems from the fact that (6.8) has to hold for *all* points from the domain Ω. A further complication is that the quantized feedback $\mu_{\rm q}(\cdot)$ is a nonlinear function (cf. (6.6) and notice that the function is nonlinear due to presence of IF-THEN logic rules), and $\underline{V}(\cdot)$ with $\overline{V}(\cdot)$ can be nonlinear functions as well.

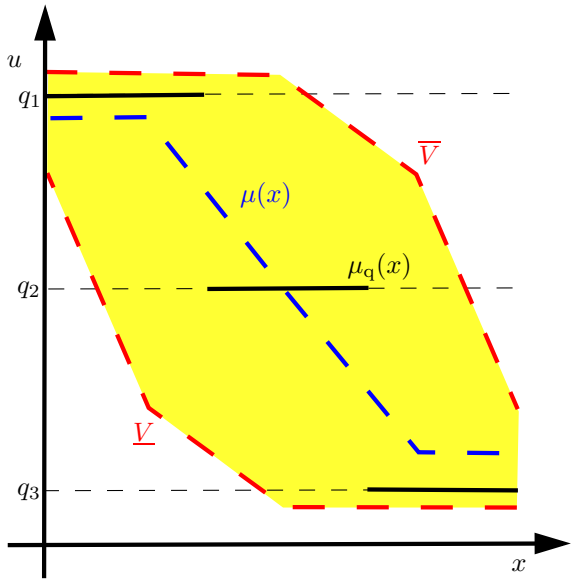


Figure 6.1: Illustrative example for the Problem 6.1.1. Here q_i (black dashed line) denotes quantization levels, $\mu(x)$ (blue dashed line) is the real-valued function and $\mu_{\mathrm{q}}(x)$ (black line) is its quantized version. Boundaries $\underline{V}/\overline{V}$ (red dashed lines) restrict the admissible area denoted by yellow color.

The condition under which we will provide an answer to Problem 6.1.1 in a non-conservative manner is summarized next.

Assumption 6.1.2 *We assume that the performance bounds $\underline{V}(\cdot)$ and $\overline{V}(\cdot)$ are piecewise affine functions*

$$\underline{V}(x) := \underline{\alpha}_{i,1}x + \underline{\alpha}_{i,0} \text{ if } x \in \mathcal{R}_i, \tag{6.9a}$$

$$\overline{V}(x) := \overline{\alpha}_{i,1}x + \overline{\alpha}_{i,0} \text{ if } x \in \mathcal{R}_i, \tag{6.9b}$$

with $\alpha_{i,1} \in \mathbb{R}^{m\times n}$, $\alpha_{i,0} \in \mathbb{R}^m$, and the polyhedra $\mathcal{R}_i$, $i = 1, \ldots, M$ are the same as in (6.5).

Remark 6.1.3 *The assumption that polyhedra over which* (6.9) *is defined are identical to those of $\mu(\cdot)$ in* (6.5) *is not restrictive as long as* $\mathrm{dom}(\underline{V}) = \mathrm{dom}(\overline{V}) = \mathrm{dom}(\mu)$ *is assumed. If $V(\cdot)$ of* (6.9) *is defined over polyhedra $\overline{\mathcal{R}}_1, \ldots, \overline{\mathcal{R}}_{\overline{M}}$, and the feedback $\mu(\cdot)$ is piecewise affine over $\widetilde{\mathcal{R}}_1, \ldots, \widetilde{\mathcal{R}}_{\widetilde{M}}$ with $\overline{M} \neq \widetilde{M}$, then one can*

always redefine both functions over a new set of polyhedra $\mathcal{R}_{i,j} = \overline{\mathcal{R}}_i \cap \widetilde{\mathcal{R}}_j$ *for all* $i-j$ *combinations which lead to a non-empty intersection. The same procedure can be applied in the situation where* $\underline{V}(\cdot)$, $\overline{V}(\cdot)$ *are defined over different polyhedra.*

Remark 6.1.4 *Although the procedures of this method are applicable to multivariable systems with multiple inputs* ($m > 1$)*, to simplify the presentation we will henceforth assume that all functions in* (6.8) *are scalar-valued. Note that with* $m > 1$, (6.8) *can be split into a set of* m *relations where each of them has to be satisfied to certify that a vector-valued quantized feedback* $\mu_q(\cdot)$ *meets the prescribed performance bounds.*

6.1.2 Certification of Quantized Feedbacks

In this section we show how to compute a *true/false* answer to Problem 6.1.1, provided that the bounding functions $\underline{V}(\cdot)$ and $\overline{V}(\cdot)$ satisfy the condition in Assumption 6.1.2.

Analytical Form of the Quantized Feedback

We start by developing an analytical form of the quantized feedback $\mu_{\rm q}(\cdot)$ in (6.6), provided we know the analytic expression for the real-valued explicit MPC control law $\mu(\cdot)$ in (6.5). In particular, we devise regions $\mathcal{P}_1, \ldots, \mathcal{P}_D$, along with the quantization level active in each region, such that $\mu_{\rm q}(\cdot)$ is given as a piecewise affine constant function of the form

$$\mu_{\rm q}(x) := \begin{cases} c_1 & \text{if } x \in \mathcal{P}_1, \\ \quad \vdots & \\ c_D & \text{if } x \in \mathcal{P}_D. \end{cases} \tag{6.10}$$

where $c_i \in \{q_1, \ldots, q_d\}$ for $i = 1, \ldots, D$, and $\mathcal{P}_i$ are polyhedra in $\mathbb{R}^n$.

Consider the k-th polyhedron $\mathcal{R}_k$ of the real-valued feedback $\mu(\cdot)$ where $\mu(x) = F_k x + g_k$ is the local feedback law. Then the subset of $\mathcal{R}_k$ where the control action $u = \mu(x)$ is rounded towards the i-th quantization level q_i is given by

$$\mathcal{P}_{k,i} = \{x \in \mathcal{R}_k \mid \|\mu(x) - q_i\|_2 \leq \|\mu(x) - q_j\|_2,\ \forall j \neq i\}. \tag{6.11}$$

Recall that $\mathcal{R}_k$ is assumed to be a polyhedron and $\mu(x)$ is an affine function $\forall x \in \mathcal{R}_k$ by (6.5). Then $\mathcal{P}_{k,i}$ is a polyhedron. To see this, note that the inequalities that

constitute $\mathcal{P}_{k,i}$ involve non-negative functions. Hence squaring both sides does not change the sign and we obtain

$$(\mu(x) - q_i)^T(\mu(x) - q_i) \leq (\mu(x) - q_j)^T(\mu(x) - q_j), \tag{6.12}$$

which simplifies to

$$2\mu(x)^T(q_j - q_i) \leq q_j^T q_j - q_i^T q_i. \tag{6.13}$$

Note that (6.13) has to hold $\forall j \neq i$. Finally, since $\mu(x) = F_k x + g_k$ for all $x \in \mathcal{R}_k$ per (6.5), and because $q_1, \ldots, q_d$ are known and fixed, expressions in (6.13) become

$$2(F_k x + g_k)^T(q_j - q_i) \leq q_j^T q_j - q_i^T q_i, \ \forall j \neq i, \tag{6.14}$$

which are $d-1$ linear inequalities in x. Then $\mathcal{P}_{k,i}$ is given by

$$\mathcal{P}_{k,i} = \{x \mid 2(F_k x + g_k)^T(q_j - q_i) \leq q_j^T q_j - q_i^T q_i, \ \forall j \neq i\}, \tag{6.15}$$

which furthermore needs to be intersected with $\mathcal{R}_k$. More generally, $\mathcal{P}_{k,i}$ is the i-th cell of the Voronoi diagram (Aurenhammer, 1991) of the points $q_1, \ldots, q_d$, where each cell is intersected with $\mathcal{R}_k$. It follows directly from properties of Voronoi diagrams that, for a fixed k, $\text{int}(\mathcal{P}_{k,i}) \cap \text{int}(\mathcal{P}_{k,j}) = \emptyset$ and $\cup_i \mathcal{P}_{k,i} = \mathcal{R}_k$. Naturally, the sets $\mathcal{P}_{k,i}$ can be empty for some $i \in \{1, \ldots, d\}$. However, there always exists at least one index i for which $\mathcal{P}_{k,i}$ is not empty.

The analytic representation of the quantized feedback in (6.6) can be obtained per Algorithm 8. The algorithm iterates over polyhedra $\mathcal{R}_k$ which define the real-valued feedback $\mu(\cdot)$ in (6.5). Next, for each $\mathcal{R}_k$ the i-th cell of the Voronoi diagram is computed in Step 4. If the intersection of the cell with $\mathcal{R}_k$ is non-empty, it is added to the set of polyhedra $\mathcal{P}$, the counter is updated, and the "active" quantization level associated to $\mathcal{P}_{k,i}$ is recorder. Upon exit, the algorithm generates polyhedra $\mathcal{P}_1, \ldots, \mathcal{P}_D$ of $\mu_{\text{q}}(\cdot)$, along with information of which quantization level is active in each region. We remark that the upper bound on the total number of regions generated by Algorithm 8 is $D = dM$.

Certification of Bounded Performance

With the analytic form of the quantized feedback (6.10) in hand, we will next show how to formulate the certification problem (6.8). To derive the main result, we will make use of the following two intermediate statements.

Algorithm 8: Construction of the quantized feedback in (6.6).

Input: Real-valued feedback: $\mu(x) := F_k x + g_k$ if $x \in \mathcal{R}_k, k = 1, \ldots, M$,
Quantization levels: $q_i, i = 1, \ldots, d$

Output: Quantized feedback: $\mu_{\mathrm{q}}(x) := c_i$ if $x \in \mathcal{P}_i, i = 1, \ldots, D$

Initialization: $D \leftarrow 0$, $\mathcal{P} \leftarrow \{\}$, $c \leftarrow \{\}$;
for $k = 1, \ldots, M$ **do**
  for $i = 1, \ldots, d$ **do**
    Compute $\mathcal{P}_{k,i}$ per (6.15);
    if $\mathcal{P}_{k,i} \cap \mathcal{R}_k \neq \emptyset$ **then**
      $D \leftarrow D + 1$;
      $\mathcal{P} \leftarrow \mathcal{P} \cup \{\mathcal{P}_{k,i} \cap \mathcal{R}_k\}$;
      $c \leftarrow c \cup \{q_i\}$;
    end
  end
end

Proposition 6.1.5 *Let $h(x)$ be an arbitrary function of x and let $\mathcal{D}$ be a non-empty set with $\mathcal{D} \subseteq \mathrm{dom}(h)$. Then*

$$h(x) \geq 0 \ \forall x \in \mathcal{D} \tag{6.16}$$

if and only if

$$\min_{x \in \mathcal{D}} h(x) \geq 0. \tag{6.17}$$

Proof. (6.17) $\Rightarrow$ (6.16): Since $\mathcal{D}$ is assumed to be non-empty, the minimum of $\min_{x \in \mathcal{D}} h(x)$ is always attained. Denote by $x^\star$ the minimizer. Then, for any $x \in \mathcal{D}$, we have $h(x) \geq h(x^\star)$ by definition of minimum. Since $h(x^\star) \geq 0$ is assumed, it follows directly that $h(x) \geq 0 \ \forall x \in \mathcal{D}$.

(6.16) $\Rightarrow$ (6.17): Assume by contradiction that (6.16) is satisfied, but (6.17) is not, i.e., we have $h(x) \geq 0 \ \forall x \in \mathcal{D}$ and $h(x^\star) < 0$ where $x^\star$ is the minimizer of (6.17). Since $x^\star \in \mathcal{D}$ because $\mathcal{D}$ is non-empty, from (6.16) we have $h(x^\star) \geq 0$, a contradiction.

Proposition 6.1.6 *$h(x) \leq 0 \ \forall x \in \mathcal{D}$ if and only if*

$$\max_{x \in \mathcal{D}} h(x) \leq 0. \tag{6.18}$$

Proof. Follows directly from Proposition 6.1.5 since $\max(h) = -\min(-h)$.

Our first main result is that the certification of performance bounds in (6.8) can be approached by investigating the minima/maxima of the performance bound functions $\underline{V}(\cdot)$ and $\overline{V}(\cdot)$ when x is restricted to all non-empty intersections $\mathcal{P}_i \cap \mathcal{R}_j$, where $\mathcal{P}_i$ is a particular polyhedron of the piecewise constant quantized feedback $\mu_{\rm q}(\cdot)$ in (6.10), and $\mathcal{R}_j$ is a region of the piecewise affine performance bounds in (6.9).

Theorem 6.1.7 *Let the quantized feedback $\mu_q(\cdot)$ in (6.10) be given, along with piecewise affine performance bounds $\underline{V}(\cdot)$, $\overline{V}(\cdot)$ of (6.9). Furthermore, denote $v(x, \underline{\alpha}_j) = \underline{\alpha}_{j,1}x + \underline{\alpha}_{j,0}$ and $v(x, \overline{\alpha}_j) = \overline{\alpha}_{j,1}x + \overline{\alpha}_{j,0}$ for $j = 1, \ldots, M$. Then $\underline{V}(x) \leq \mu_q(x) \leq \overline{V}(x)$ for all $x \in \Omega$, i.e., (6.8) holds, if and only if*

$$\max_{x \in \mathcal{P}_i \cap \mathcal{R}_j} (c_i - v(x, \overline{\alpha}_j)) \leq 0 \leq \min_{x \in \mathcal{P}_i \cap \mathcal{R}_j} (c_i - v(x, \underline{\alpha}_j)) \tag{6.19}$$

is satisfied for all $i = 1, \ldots, D$, $j = 1, \ldots, M$ for which $\mathcal{P}_i \cap \mathcal{R}_j \neq \emptyset$.

Proof. Consider an arbitrary combination of indices i and j for which $\mathcal{Q}_{i,j} = \mathcal{P}_i \cap \mathcal{R}_j$ is non-empty and note that $\mathcal{Q}_{i,j}$ is a polyhedron, for it is an intersection of two polyhedra. For all x restricted to $\mathcal{Q}_{i,j}$ we have $\mu_{\rm q}(x) := c_i$, $\underline{V}(x) := v(x, \underline{\alpha}_j)$, and $\overline{V}(x) := v(x, \overline{\alpha}_j)$. Then $\underline{V}(x) \leq \mu_{\rm q}(x)$ for all $x \in \mathcal{Q}_{i,j}$ if and only if

$$c_i - v(x, \underline{\alpha}_j) \geq 0, \ \forall x \in \mathcal{Q}_{i,j}, \tag{6.20}$$

which is equivalent, by Proposition 6.1.5, to

$$\min_{x \in \mathcal{Q}_{i,j}} \big(c_i - v(x, \underline{\alpha}_j)\big) \geq 0, \tag{6.21}$$

and the right-hand-side of (6.19) follows. Similarly, $\mu_{\rm q}(x) \leq \overline{V}(x)$ for all $x \in \mathcal{Q}_{i,j}$ if and only if

$$c_i - v(x, \overline{\alpha}_j) \leq 0, \ \forall x \in \mathcal{Q}_{i,j}, \tag{6.22}$$

which, by Proposition 6.1.6, holds if and only if

$$\max_{x \in \mathcal{Q}_{i,j}} \big(c_i - v(x, \overline{\alpha}_j)\big) \leq 0, \tag{6.23}$$

and we get the left-hand-side of (6.19). Repeating the argument for all $i = 1, \ldots, D$ (i.e., for all regions of the piecewise constant quantized feedback in (6.10)) and $j = 1, \ldots, M$ (where M is the number of regions of $\underline{V}(\cdot)$ and $\overline{V}(\cdot)$) yields the

theorem. Note that $\cup_{i,j}(\mathcal{R}_i \cap \mathcal{P}_j) = \cup_i \cup_j (\mathcal{R}_i \cap \mathcal{P}_j) = (\cup_i \mathcal{R}_i) \cap (\cup_j \mathcal{P}_j) = \Omega \cap \Omega = \Omega$, therefore the procedure covers all $x \in \Omega$ where Ω is the domain of $\underline{V}(\cdot)$, $\overline{V}(\cdot)$ and $\mu_{\rm q}(\cdot)$.

Remark 6.1.8 *Note that the optimization problems in* (6.19) *are always feasible since we assume that x is constrained to belong to a non-empty set $\mathcal{P}_i \cap \mathcal{R}_j$.*

To exploit Theorem 6.1.7 to certify satisfaction of (6.8) for *all* $x \in \Omega$ we thus need to solve up to DM problems (6.19), each of which involves solving two optimization problems. In practice, the number will be less, since only non-empty intersections $\mathcal{P}_i \cap \mathcal{R}_j$ need to be considered. The complete certification procedure is reported as Algorithm 9. Checking whether $\mathcal{P}_i \cap \mathcal{R}_j = \emptyset$ in Step 3 can be performed at the cost of solving one linear program, see e.g. (Borrelli, 2003a).

Algorithm 9: Certification of performance bounds in (6.8).

Input: $\mu_{\rm q}(\cdot)$, $\overline{V}(\cdot)$, $\underline{V}(\cdot)$

Output: *true/fasle* answer for (6.8)

```
1  for i = 1, ..., D do
2      for j = 1, ..., M do
3          if P_i ∩ R_j ≠ ∅ then
4              Solve the maximization/minimization problems in (6.19);
5              if (6.19) is not satisfied then
6                  return (6.8) not satisfied;
7              end
8          end
9      end
10 end
11 return (6.8) satisfied ∀x ∈ dom(μ_q);
```

In a more general sense, Theorem 6.1.7 provides necessary and sufficient conditions for satisfaction of (6.8) for arbitrary bounding functions $\underline{V}(\cdot)$, $\overline{V}(\cdot)$, e.g. for piecewise quadratic or piecewise polynomial functions. Then, however, the main difficulty there is that the optimization problems in (6.19) can be non-convex. Determining globally optimal solutions to non-convex problems is computationally challenging. With $\underline{V}(\cdot)$, $\overline{V}(\cdot)$ restricted to piecewise affine functions, on the other

hand, the problems in (6.19) are linear programs that can be solved in polynomial time. To see this, note that the objective functions are linear in x, and the constraints are linear as well, since $\mathcal{P}_i \cap \mathcal{R}_j$ are polyhedra. Therefore the problems in (6.19) can be solved even in large dimensions with off-the-shelf optimization packages, such as CPLEX or GUROBI, or even with open-source alternatives such as GLPK or CDD.

6.1.3 Construction of Performance Bounds

In this section we present construction of performance bounds $\underline{V}(\cdot)$, $\overline{V}(\cdot)$ in (6.8) that reflect typical control requirements, such as recursive satisfaction of state/input constraints, closed-loop stability or bounded deterioration of control performance. We focus on construction of piecewise affine bounds $\underline{V}(\cdot)$, $\overline{V}(\cdot)$ for which validity of (6.8) can be certified in a non-conservative manner by solving a series of linear programs in (6.19). The development will be based on the assumption that the constraints in (6.2) are polytopic.

Recursive Satisfaction of State and Input Constraints

For the system in (6.1) with constraints as in (6.2), the positive control invariant set is given by

$$\mathcal{C} = \{x_0 \in \mathcal{X} \mid \exists \kappa : Ax_k + B\kappa(x_k) \in \mathcal{X},\ \kappa(x_k) \in \mathcal{U}, \forall k \in \mathbb{N}\}, \tag{6.24}$$

where $\kappa : \mathbb{R}^n \to \mathbb{R}^m$ is a feedback strategy, and x_k and u_k are the states and inputs at the discrete time step k. Under mild conditions, the set $\mathcal{C}$ is a polytope and can be computed for instance by Multi-Parametric Toolbox (Herceg et al., 2013a). The important property of the control invariant set is that for any $x \in \mathcal{C}$ there always exist a control action $u \in \mathcal{U}$ that keeps the state update inside of the state constraints, i.e., $Ax + Bu \in \mathcal{X}$ for all time. Let us denote by

$$\mathcal{C}_{\text{xu}} = \{(x, u) \mid x \in \mathcal{C},\ u \in \mathcal{U},\ Ax + Bu \in \mathcal{C}\} \tag{6.25}$$

the set of all state-input pairs for which the state update $Ax + Bu \in \mathcal{C}$. Therefore for any x and u satisfying $(x, u) \in \mathcal{C}_{\text{xu}}$ we have that: 1) $x \in \mathcal{X}$, 2) $u \in \mathcal{U}$, and 3) $Ax + Bu \in \mathcal{X}$. Therefore if u is selected such that $(x, u) \in \mathcal{C}_{\text{xu}}$, recursive satisfaction of state and input constraints is enforced. Since $\mathcal{C}_{\text{xu}}$ is a polytope, it

can be represented by

$$\mathcal{C}_{\text{xu}} = \{(x, u) \mid Gx + Hu \leq h\}. \tag{6.26}$$

Then the bounds $\underline{V}(\cdot)$, $\overline{V}(\cdot)$ that reflect recursive satisfaction of state and input constraints can be computed as follows:

$$\underline{V}(x) = \{\min u \mid Gx + Hu \leq h\}, \tag{6.27a}$$

$$\overline{V}(x) = \{\max u \mid Gx + Hu \leq h\}. \tag{6.27b}$$

The problems in (6.27) are *parametric* linear programs with optimization variables $u \in \mathbb{R}^m$ and parameters $x \in \mathbb{R}^n$:

Lemma 6.1.9 (Borrelli (2003b)) *The solution to* (6.27) *are PWA functions*

$$\underline{V}(x) := \underline{\alpha}_{i,1}x + \underline{\alpha}_{i,0} \text{ if } x \in \mathcal{R}_i, \tag{6.28a}$$

$$\overline{V}(x) := \overline{\alpha}_{i,1}x + \overline{\alpha}_{i,0} \text{ if } x \in \mathcal{R}_i, \tag{6.28b}$$

where $\mathcal{R}_i$ are non-overlapping polytopes[2] *in $\mathbb{R}^n$.*

To construct $\underline{V}(\cdot)$, $\overline{V}(\cdot)$ in (6.28), one needs to solve the parametric linear programs (6.27). This can be achieved e.g. by the Multi-Parametric Toolbox.

Remark 6.1.10 *The worst-case complexity of parametric linear programs in* (6.27)*, i.e., the upper bound on the number of polytopes $\mathcal{R}_i$ in* (6.28)*, is exponential in the number of constraints, which in turn depends on dimensions of u and x. However, recent development (Gupta et al., 2011) in the theory of parametric optimization allows to solve such problems even in large dimensions, say over* 50 *(cf. Section 4.2).*

Closed-Loop Stability

Let a piecewise linear convex Lyapunov function $V : \mathbb{R}^n \to \mathbb{R}$ for the system in (6.1), i.e.,

$$V(x) := \max_i \; w_i^T x \tag{6.29}$$

with $w_i \in \mathbb{R}^n$, $i = 1, \ldots, W$ be given. Then any feedback policy $\kappa(x)$ that guarantees

$$V(Ax + B\kappa(x)) \leq \gamma V(x), \; \forall x \in \text{dom}(\kappa), \tag{6.30}$$

[2]The polytopes in (6.28a) and (6.28b) are the same since both problems in (6.27) have identical constraints.

for some $0 \leq \gamma < 1$ trivially provides closed-loop stability as it forces the Lyapunov function to decrease along the state trajectories, see e.g. Lazar et al. (2008). The minimal and maximal control actions, as a function of the states x, which render the closed-loop stable in the sense of (6.30) can thus be obtained by

$$\underline{V}(x) = \{\min u \mid V(Ax+Bu) \leq \gamma V(x),\ u \in \mathcal{U}\}, \tag{6.31a}$$

$$\overline{V}(x) = \{\max u \mid V(Ax+Bu) \leq \gamma V(x),\ u \in \mathcal{U}\}. \tag{6.31b}$$

With $V(\cdot)$ as in (6.29), problems (6.31) can be formulated and solved as *parametric mixed-integer linear programs* in decision variables u and parameters x by introducing additional binary variables to model the maxima in (6.29) as follows:

Proposition 6.1.11 (Kvasnica et al. (2012)) *The maximum among linear functions of x, i.e., $t = \max_i w_i^T x$, is modeled by*

$$-M(1-\delta_i) \leq t - w_i^T x \leq M(1-\delta_i), \tag{6.32a}$$

$$w_j^T x \leq w_i^T x + M(1-\delta_i),\ \forall j \neq i, \tag{6.32b}$$

$$\textstyle\sum_{i=1}^{V} \delta_i = 1, \tag{6.32c}$$

where $\delta_i \in \{0,1\}$, $i = 1,\ldots,W$ are binary variables, M is a sufficiently large positive constant, and (6.32a)−(6.32b) *are enforced for all $i = 1,\ldots,W$.*

With the help of Proposition 6.1.11, problem (6.31a) can be rewritten as

$$\underline{V}(x) = \{\min u \mid y \leq \gamma t,\ u \in \mathcal{U}\}, \tag{6.33}$$

where t is related to $\max_i w_i^T x$ via (6.32), and y models $\max_i w_i^T(Ax+Bu)$ via the same relations. Note that when $V(\cdot)$ in (6.29) is a piecewise linear function of x, $V(Ax+Bu)$ is a piecewise linear function of x and u.

As shown in (Borrelli, 2003a), the solutions $\underline{V}(x)$ and $\overline{V}(x)$ to (6.31) are piecewise affine functions of x as in (6.28). The local affine expressions, as well as the associated regions of validity, can be obtained by the Multi-Parametric Toolbox.

Remark 6.1.12 *One straightforward way to obtain a piecewise linear Lyapunov function in* (6.29) *is to consider the Minkowski function of a polytopic contractive set $\mathcal{C}$. Let the normalized*[3] *half-space representation of $\mathcal{C}$ be given by $\mathcal{C} = \{x \mid w_1^T x \leq 1,\ldots,w_W^T x \leq 1\}$, where W is the number of facets of $\mathcal{C}$. Then $V(\cdot)$ of* (6.29) *is a Lyapunov function (Blanchini and Miani, 2008).*

[3]Every polyhedron that contains the origin in its interior can be normalized by $\mathcal{P} = \{x \mid Lx \leq \mathbf{1}\}$ where $\mathbf{1} = (1,\ldots,1)$.

Bounded Performance Deterioration

Assume a performance measure $J : \mathbb{R}^n \to \mathbb{R}$, $J(x) \geq 0 \ \forall x \in \mathrm{dom}(J)$, given by

$$J_\kappa(x) := \|Ax + B\kappa(x)\|_\infty, \tag{6.34}$$

which measures how close, in the ∞-norm, to the origin a particular state x can be pushed by the feedback $u = \kappa(x)$. Then one of the possible performance bounds that we might require the quantized feedback $\mu_{\mathrm{q}}(\cdot)$ to satisfy is given by

$$J_{\mu_{\mathrm{q}}}(x) \leq \omega J_\mu(x), \ \forall x \in \mathrm{dom}(\mu), \tag{6.35}$$

where $\mu(\cdot)$ is the real-valued explicit MPC feedback in (6.5), and $\omega > 1$ is given. Put simply, we want to verify that the performance of the quantized feedback $\mu_{\mathrm{q}}(\cdot)$ is by at most a ω factor worse than the performance of the real-valued policy $\mu(\cdot)$. To translate (6.35) into (6.8), we proceed as follows. Denote by $\mathcal{S}_{\mathrm{xu}}$ the set of (x, u) vectors for which (6.35) holds, i.e.,

$$\mathcal{S}_{\mathrm{xu}} = \{(x, u) \mid \|Ax + Bu\|_\infty \leq \omega(\|Ax + B\mu(x)\|_\infty, \ u \in \mathcal{U}\}. \tag{6.36}$$

Then the performance bounds $\underline{V}(\cdot)$ $\overline{V}(\cdot)$ that entail (6.35) are

$$\underline{V}(x) = \{\min u \mid (x, u) \in \mathcal{S}_{\mathrm{xu}}\}, \tag{6.37a}$$

$$\overline{V}(x) = \{\max u \mid (x, u) \in \mathcal{S}_{\mathrm{xu}}\}. \tag{6.37b}$$

Since $\|t\|_\infty = \max_i |t_i|$, and because $\mu(\cdot)$ is assumed to be a piecewise affine function (cf. (6.5)), the set $\mathcal{S}_{\mathrm{xu}}$ in (6.36) can be rewritten into a set of inequalities using extra binary variables as in (6.32). Hence problems (6.37) are parametric mixed integer linear programs for which the optimal solution (the lower and upper bounds $\underline{V}(\cdot)$ and $\overline{V}(\cdot)$) can be obtained by parametric optimization, similarly to the procedure of Section 6.1.3.

Remark 6.1.13 *We should note, that the verification of $\mu_q(\cdot)$, can be done sequentially. To proceed, for a particular $\mathcal{R}_j \in \mathrm{dom}(\mu(\cdot))$ one can compute $\underline{V}_{\mathcal{R}_j}(\cdot)$ and $\overline{V}_{\mathcal{R}_j}(\cdot)$, respectively, and apply the Algorithm 9 for all $j = 1, \ldots, M$. In another words, boundaries in* (6.37) *can be constructed per each affine part $F_j x + g_j$ of $\mu(x)$ in* (6.36).

6.1.4 Examples

Illustrative 1D Example

To illustrate the procedure, consider the system $x(t+1) = Ax(t) + Bu(t)$ with $A = 0.9$, $B = 1$, $\mathcal{X} = \{x \,|\, -5 \leq x \leq 5\}$, $\mathcal{U} = \{u \,|\, -1 \leq u \leq 1\}$. The system is controlled by the state-feedback regulator $u = Kx$ with $K = -0.2$. It is easy to verify that such a feedback achieves recursive satisfaction of input and state constraints and attains closed-loop stability.

Next, we investigate whether constraint satisfaction, closed-loop stability, and bounded performance deterioration are achieved by a-posteriori quantizing the real-valued feedback $\mu(x) := Kx$ using three quantization levels: $q_1 = -0.9$, $q_2 = 0$, $q_3 = 0.8$. To do so, we have first derived the analytical form of $\mu_{\rm q}(\cdot)$ in (6.10). This was achieved by applying Algorithm 8, which produced

$$\mu_{\rm q}(x) := \begin{cases} 0.8 & \text{if } -5 \leq x \leq -2, \\ 0 & \text{if } -2 \leq x \leq 2.25, \\ -0.9 & \text{if } 2.25 \leq x \leq 5 \end{cases} \tag{6.38}$$

after 0.02 seconds[4] . The real-valued feedback $\mu(\cdot)$, as well as its quantized version (6.38), are shown in Figure 6.2.

To verify whether $\mu_{\rm q}(\cdot)$ of (6.38) maintains recursive satisfaction of input and state constraints, we have proceed as in Section 6.1.3. First, the control invariant set $\mathcal{C} = \{x \,|\, -5 \leq x \leq 5\}$ was calculated by (6.24). Subsequently, the set $\mathcal{C}_{\rm xu}$ in (6.26) was constructed and represented as a polyhedron per (6.27) with

$$\mathcal{C}_{\rm xu} = \left\{ (x,u) \;\middle|\; \begin{bmatrix} H_c \\ 0 \\ H_c A \end{bmatrix} x + \begin{bmatrix} 0 \\ H_u \\ H_c B \end{bmatrix} u \leq \begin{bmatrix} h_c \\ h_u \\ h_c \end{bmatrix} \right\}, \tag{6.39}$$

where $H_c = [1,1]^T$, $h_c = [5,5]^T$ is the half-space representation of $\mathcal{C} = \{x \,|\, H_c x \leq h_c\}$, and $H_u = [1,1]^T$, $h_u = [1,1]^T$ is the half-space representation of $\mathcal{U} = \{u \,|\, H_u u \leq h_u\}$. Finally, the bounding functions $\underline{V}(\cdot)$, $\overline{V}(\cdot)$ were computed by

[4] All calculations in this section were performed on a 1.7 GHz CPU using the Multi-Parametric Toolbox 3.1.2.

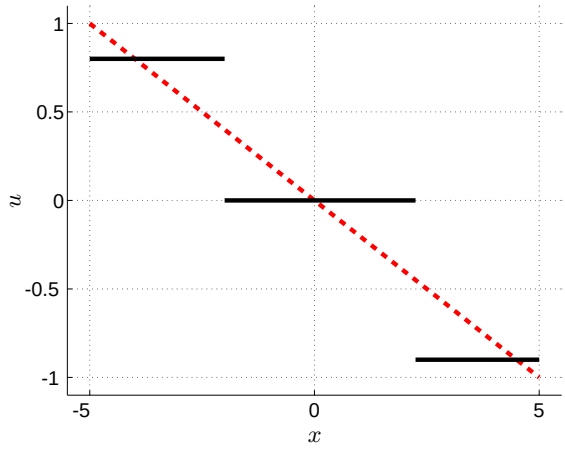


Figure 6.2: Real-valued feedback $u = Kx$ (dashed red line) and its quantized version $u = \mu_{\rm q}(x)$ in (6.38) (solid black function) for the example in Section 6.1.4.

solving (6.27) as parametric linear programs. After 0.57 seconds we have obtained

$$\underline{V}(x) := \begin{cases} -0.9x - 5 & \text{if } -5 \le x \le -4.444, \\ -1 & \text{if } -4.444x \le x \le 5, \end{cases} \tag{6.40a}$$

$$\overline{V}(x) := \begin{cases} 1 & \text{if } -5 \le x \le 4.444, \\ -0.9x + 5 & \text{if } \quad 4.444 \le x \le 5. \end{cases} \tag{6.40b}$$

The set $\mathcal{C}_{\rm xu}$, along with functions $\underline{V}(\cdot)$, $\overline{V}(\cdot)$, and $\mu_{\rm q}(\cdot)$, are shown in Figure 6.3. As can be observed, the value of $\mu_{\rm q}(x)$ is always between $\underline{V}(x)$ and $\overline{V}(x)$ for all $x \in \mathcal{C}$. Hence $\mu_{\rm q}(\cdot)$ of (6.38) provides recursive satisfaction of state and input constraints. To give a rigorous certificate that such an observation is indeed true, we have applied Algorithm 9 to $\mu_{\rm q}(\cdot)$, $\underline{V}(\cdot)$ and $\overline{V}(\cdot)$. By solving a total of 8 linear programs in (6.19), which took 0.04 seconds, a positive certificate for Problem 6.1.1 was indeed obtained.

Then we have investigated whether $\mu_{\rm q}(\cdot)$ in (6.38) provides closed-loop stability. To do so, we have first constructed functions $\underline{V}(\cdot)$ and $\overline{V}(\cdot)$ by solving parametric mixed-integer problems in (6.31) with the Lyapunov function given as the Minkowski function of the control invariant set $\mathcal{C}$. The piecewise affine functions were constructed in 0.81 seconds and each of them was defined over 4 polyhedra.

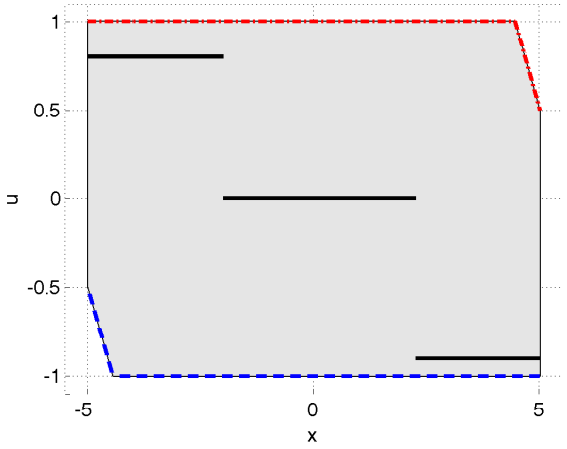


Figure 6.3: Results for feasibility verification in Section 6.1.4: the set $\mathcal{C}_{\mathrm{xu}}$ from (6.39) in gray, $\underline{V}(\cdot)$ in (6.40a) as the blue dashed function, $\overline{V}(\cdot)$ from (6.40b) as the red dash-dotted function, and the quantized feedback in (6.38) as the black function.

The resulting bounding functions, along with $\mu_{\mathrm{q}}(\cdot)$, are shown in Figure 6.4. As can be observed from the figure, all values of $\mu_{\mathrm{q}}(x)$ are always bounded by $\underline{V}(x)$ and $\overline{V}(x)$, hence the quantized feedback (6.38) provides closed-loop stability. A rigorous certificate of such a property was obtained by applying Algorithm 9, which required solving 12 linear programs for 12 non-empty intersections $\mathcal{P}_i \cap \mathcal{R}_j$.

Finally, to verify whether or not the a-posterior quantized feedback (6.38) provides bounded deterioration of performance, we have constructed the bounding functions $\underline{V}(\cdot)$, $\overline{V}(\cdot)$ per Section 6.1.3 with $\omega = 1.05$ in (6.36). This setting corresponds to verifying that $\mu_{\mathrm{q}}(\cdot)$ performs by at most 5 % worse than the real-valued controller $\mu(\cdot)$. The corresponding piecewise affine bounding functions were constructed by solving the parametric mixed integer linear programs in (6.37), which took 0.80 seconds. The results are plotted in Figure 6.5. As can be observed, $\mu_{\mathrm{q}}(\cdot)$ is *not* always between $\underline{V}(\cdot)$ and $\overline{V}(\cdot)$. Since Theorem 6.1.7 is non-conservative for piecewise affine bounds, we conclude that $\mu_{\mathrm{q}}(\cdot)$ *does not* provide ω-deterioration of closed-loop performance. Such a conclusion was verified by Algorithm 9, which has found a violation of (6.19) for the middle region of $\mu_{\mathrm{q}}(\cdot)$ where $u = 0$. As an example, consider $x = 3$, for which $\mu(x) = -0.6$. Then $Ax + B\mu(x) = 2.1$. With the

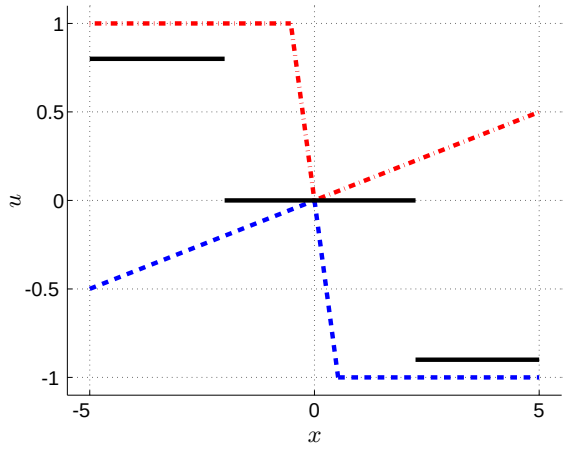


Figure 6.4: Results for stability verification in Section 6.1.4: functions $\underline{V}(\cdot)$ (blue dashed), $\overline{V}(\cdot)$ (red dash-dotted) from (6.31), and the quantized feedback $\mu_{\mathrm{q}}(\cdot)$ in (6.38).

quantized feedback, $\mu_{\mathrm{q}}(x) = -0.9$, which gives $Ax + B\mu_{\mathrm{q}}(x) = 1.8$, which is within ω-factor increase of $\|2.1\|_\infty$. However, for $x = 1$ we have $Ax + B\mu(x) = 0.7$, and $Ax + B\mu_{\mathrm{q}}(x) = 0.9$. Here, $0.9 \not\leq \omega 0.7$ with $\omega = 1.05$. All results of the verification are compactly reported in Table 6.1.

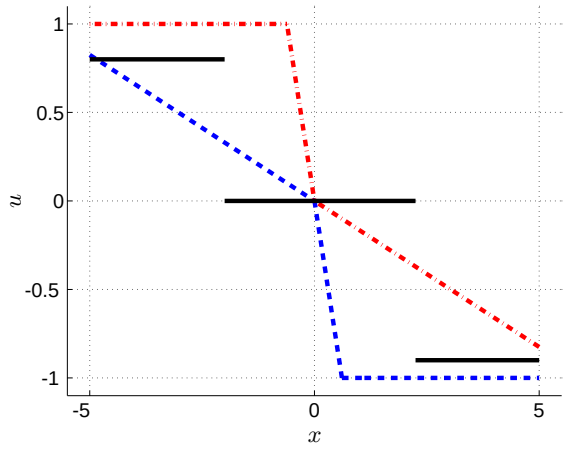


Figure 6.5: Results for performance verification in Section 6.1.4: functions $\underline{V}(\cdot)$ (blue dashed), $\overline{V}(\cdot)$ (red dash-dotted) from (6.37), and the quantized feedback $\mu_{\mathrm{q}}(\cdot)$ in (6.38).

	Operation	Time $[s]$	Complexity	Result
Quantization	$\mu(x)$	0.02	1 region	
	$\mu_{\mathrm{q}}(x)$	0.02	3 regions	
Feasibility	$\overline{V}/\underline{V}$	0.57	2 regions	*Passed*
	Certification	0.04	8 LPs	
Stability	$\overline{V}/\underline{V}$	0.81	4 regions	*Passed*
	Certification	0.07	12 LPs	
Performance	$\overline{V}/\underline{V}$	0.80	4 regions	*Failed*
	Certification	0.03	5 LPs	

Table 6.1: Results of the verification for 1D example.

Ball and the Beam

Next we consider a system of ball on the beam, described in Figure 6.6. The nonlinear dynamics was linearized around the beam's angle $\alpha = 0$, what yielded the linear model in the form:

$$\begin{bmatrix} \dot{r} \\ \ddot{r} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} r \\ \dot{r} \end{bmatrix} + \begin{bmatrix} 0 \\ -0.21 \end{bmatrix} \theta, \tag{6.41}$$

where $|r| \leq 0.5$ is the position of the ball, $|\dot{r}| \leq 1$ is the speed of the ball and with the servo gear angle $|\theta| \leq 1.5$ representing the control input to the system.

To obtain model as in (6.1), we have discretized the system (6.41) with sampling time equal to 0.1 seconds. Then, the MPC problem in (6.3) was formulated with $N = 5$, weighting matrices $Q_x = \mathrm{diag}(1, 0, 1)$, $Q_u = 0.3$, Q_N equal to the solution of the algebraic Riccati equation, and $\mathcal{X}_f$ being the constraint LQR terminal set. By solving the given problem for all feasible initial conditions resulted in the explicit MPC policy $\mu(\cdot)$ defined over 37 regions.

To analyze the performance of $\mu(\cdot)$ under the rounding-based quantizer with 13 equidistantly distributed quantization levels, i.e. $\{-1.5, -1.25, -1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1, 1.25, 1.5\}$, we have devised the $\mu_{\mathrm{q}}(\cdot)$ as in (6.10) per Algorithm 8. After 1.33 seconds, this resulted in a piecewise constant feedback law $\mu_{\mathrm{q}}(\cdot)$ defined over 55 regions.

Firstly, we have verified, whether the a-posteriori quantized controller $\mu_{\mathrm{q}}(\cdot)$ provides the recursive feasibility for an arbitrary controllable initial condition. To

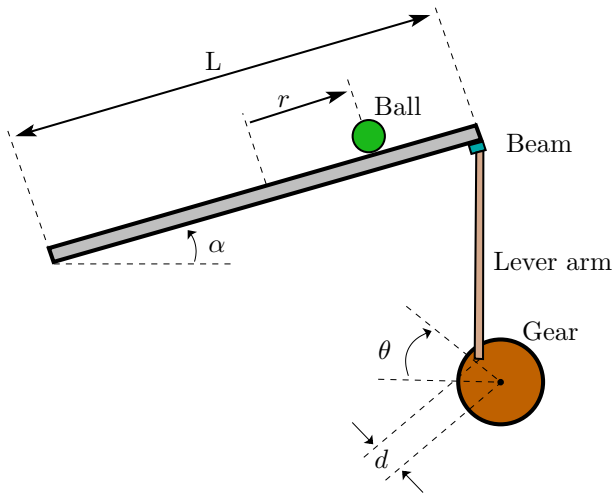


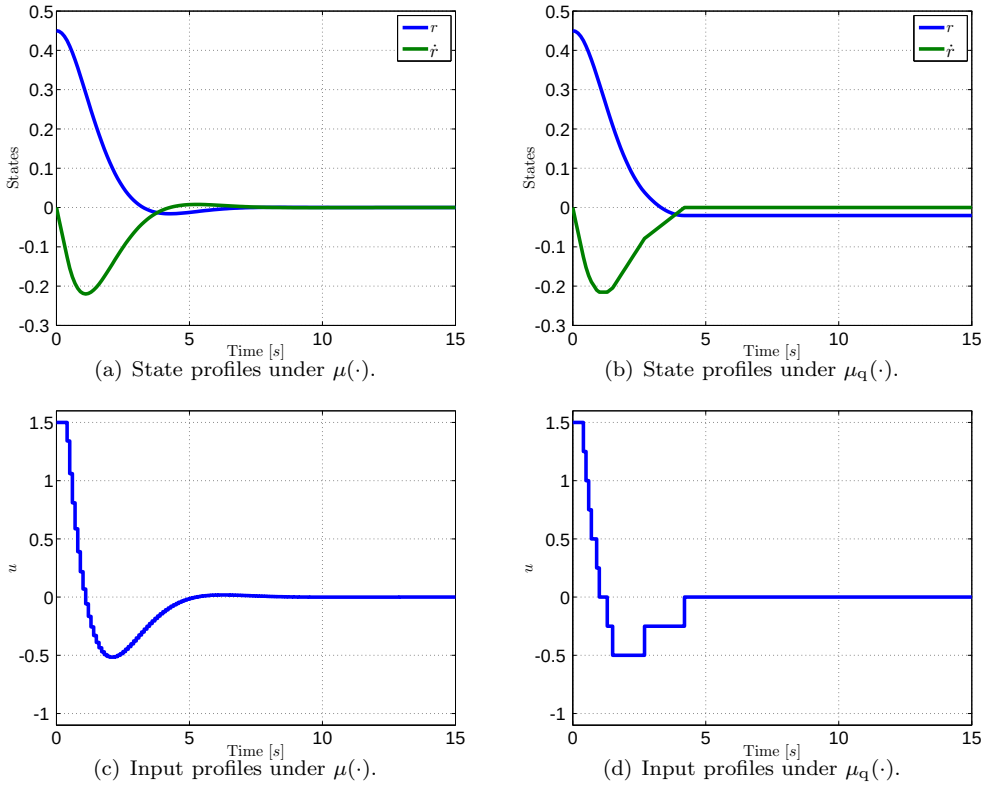
Figure 6.6: Ball and the beam.

proceed, we have constructed performance boundaries $\underline{V}(\cdot)$ and $\overline{V}(\cdot)$ as described in Section 6.1.3. The computation of both functions, along with $\mathcal{C}$, took 2.92 seconds and their polytopic partition $\mathrm{dom}(\underline{V}(\cdot))$ and $\mathrm{dom}(\overline{V}(\cdot))$ were composed of 7 regions each. Then, the Algorithm 9 was employed to determine validity of (6.8). In 2.45 seconds a positive certificate was obtained, thus we have that $\mu_{\mathrm{q}}(\cdot)$ satisfies the input and state constraints.

Subsequently, we have investigated if $\mu_{\mathrm{q}}(\cdot)$ exhibits property of closed-loop asymptotic stability. For this, boundaries in (6.31) were devised in 12.84 seconds and employed in Algorithm 4, which (after 0.25 seconds) yielded a negative certificate. To see this, in Figure 6.7, we have shown closed-loop simulation of $\mu(\cdot)$ and its quantized version $\mu_{\mathrm{q}}(\cdot)$, for the initial condition $x = [0.1\ 0]^T$. Here, one can see that the a-posteriori quantized controller $\mu_{\mathrm{q}}(\cdot)$ does not steer all states to the origin. This is due to the fact that, at the time $t = 4.2$ seconds, the control input from the MPC policy $\mu(\cdot)$ was rounded to the nearest quantization level $q_6 = 0$. By plugging $\mu_{\mathrm{q}}(\cdot) = 0$ into the systems dynamics in (6.1), we have $x(t+1) = x(t)$, and this applies for all further time steps $t > 4.2$. Finally, since $\lim_{t\to\infty} x(t) \neq 0$, we have that $\mu(\cdot)$ does not exhibit closed-loop asymptotic stability[5], what proofs the result of the obtained certificate.

In the end, we have determined whether performance decadency of $\mu_{\mathrm{q}}(\cdot)$ is not

[5]Thought it provides bounded-input bounded-output (BIBO) stability



(a) State profiles under $\mu(\cdot)$.

(b) State profiles under $\mu_{\mathrm{q}}(\cdot)$.

(c) Input profiles under $\mu(\cdot)$.

(d) Input profiles under $\mu_{\mathrm{q}}(\cdot)$.

Figure 6.7: Simulation results for the system ball and the beam. Left column shows state and input profiles when the system is controlled by the real-valued MPC feedback. The column on the right corresponds to profiles obtained by controlling the system by the quantized feedback. The initial condition was $r = 0.45$, $\dot{r} = 0$.

larger then 30%, compared to the real-valued controller $\mu(\cdot)$. Here, a sequential approach was applied, where (6.8) was iteratively verified with boundary functions $\underline{V}_j(\cdot)$ and $\overline{V}_j(\cdot)$ constructed for each region $j = 1, \ldots, M$. Since, after 111 seconds no violation was detected, we have that $\mu_{\mathrm{q}}(\cdot)$ provides bounded deterioration of performance for $\gamma = 1.3$.

Results of all certification procedures are reported in Table 6.2.

	Operation	Time [s]	Complexity	Result
Quantization	$\mu(x)$	3.38	37 regions	
	$\mu_{\mathrm{q}}(x)$	1.33	55 regions	
Feasibility	$\overline{V}/\underline{V}$	2.92	7 regions	*Passed*
	Certification	2.45	120 LPs	
Stability	$\overline{V}/\underline{V}$	12.84	57 regions	*Failed*
	Certification	0.25	19 LPs	
Performance	$\overline{V}/\underline{V}$	86.35	74 regions[6]	*Passed*
	Certification	23.71	284 LPs	

Table 6.2: Results of the verification for ball and the beam.

Inverted Pendulum on a Cart

As the last example we will use an inverted pendulum mounted on a moving cart, shown in Figure 5.3 in Section 5.7.2. Linearizing the nonlinear dynamics around the upright, marginally stable position leads to the following linear model:

$$\begin{bmatrix} \dot{p} \\ \ddot{p} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -0.182 & 2.673 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -0.455 & 31.182 & 0 \end{bmatrix} \begin{bmatrix} p \\ \dot{p} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 1.818 \\ 0 \\ 4.546 \end{bmatrix} u, \tag{6.42}$$

where p is the position of the cart (constrained by $|p| \leq 1$), $\dot{p}$ is the cart's velocity (with $|\dot{p}| \leq 1$), ϕ is the pendulum's angle from the upright position (with $|\phi| \leq 0.35$), and $\dot{\phi}$ denotes the angular velocity (restricted to $|\dot{\phi}| \leq 1$). The control input u, constrained to $|u| \leq 1$, is proportional to the force applied to the cart. System (6.42) was converted to (6.1) by assuming sampling time 0.1 seconds. For the discrete-time system we have first constructed the real-valued feedback in (6.5) by solving the MPC problem in (6.3) with the prediction horizon $N = 4$, and penalties $Q_{\mathrm{x}} = \mathrm{diag}(10, 1, 10, 1)$, $Q_{\mathrm{u}} = 0.1$. Moreover, the terminal penalty Q_{N} was selected as the solution to the algebraic Riccati equation, while $\mathcal{X}_f$ is the LQR terminal set. By solving (6.3) parametrically, the feedback $\mu(x)$ was obtained as a piecewise affine function defined over 187 polytopes in $\mathbb{R}^4$.

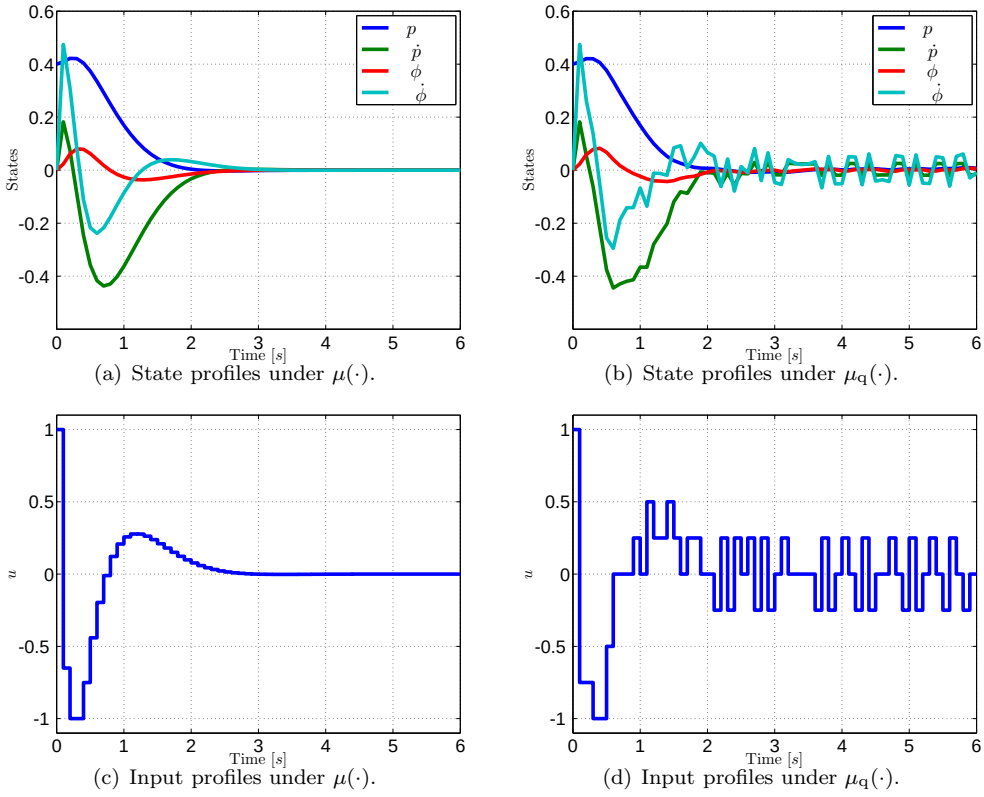
Then we have investigated the properties of the quantized version of $\mu(\cdot)$ by assuming quantization levels $\{-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1\}$. Here, the

quantized controller $\mu_{\rm q}(\cdot)$ in (6.10) was first obtained by applying Algorithm 8 to $\mu(\cdot)$. After 3.97 seconds we have obtained the quantized feedback in (6.10) which was defined over 557 polytopes. A comparison of closed-loop performance of the real-valued MPC feedback $\mu(\cdot)$ versus its a-posteriori quantized version $\mu_{\rm q}(\cdot)$ is shown in Figure 6.8.

Subsequently, we have applied the procedure of Section 6.1.3 to check whether $\mu_{\rm q}(\cdot)$ achieves recursive satisfaction of state and input constraints for an arbitrary controllable initial condition. The computation of the invariant set $\mathcal{C}$ in (6.24), along with construction of the performance bounds per (6.28), took 39.51 seconds in total. Here, the functions $\underline{V}(\cdot)$, $\overline{V}(\cdot)$ were both defined over 36 regions. Finally, Algorithm 9 was executed to check validity of (6.8). A negative certificate was obtained in 0.46 seconds after 29 LPs (out of 2864 candidates). Therefore we deduce that the a-posteriori quantized feedback $\mu_{\rm q}(\cdot)$ does not exhibit state/input constraint satisfaction. To see this we can exploit result of the certification that reports that the first violation has occurred in the intersection of the 4-th partition of $\mu_{\rm q}(x) = -0.25$ with the 22-th partition of $\overline{V}(x) = -0.27$, where $x = [-0.5108, 1.0000, 0.2907, -1.0000]^T$. Since $\overline{V}(x) < \mu_{\rm q}(x)$ we have that $\mu_{\rm q}(x)$ violates the upper boundary at least in this particular state.

As can be observed from Figure 6.8(b), for at least one initial condition the quantized feedback does not push all system's states asymptotically to the origin. Therefore $\mu_{\rm q}(\cdot)$ does not provide guarantees of asymptotic closed-loop stability . This conclusion was verified per the procedure of Section 6.1.3. Here, we have first constructed the bounding functions by solving the parametric mixed-integer linear programs in (6.31). After 761 seconds we have obtained piecewise affine functions $\underline{V}(\cdot)$ and $\overline{V}(\cdot)$ defined over 714 polytopes in $\mathbb{R}^4$. The subsequent execution of Algorithm 9 took 0.06 seconds to find a violation of (6.19), hence certifying a negative answer to verification of closed-loop stability properties.

Finally, we have verified whether $\mu_{\rm q}(\cdot)$ provides bounded deterioration of performance for $\gamma = 1.3$, what corresponds to determining, whether the performance decay of $\mu_{\rm q}(\cdot)$ is not worse than 30% compared to $\mu(\cdot)$. To proceed, we have employed sequential verification (cf. Remark 6.1.13), which after 13 seconds yielded the negative certificate. Particularly, the violation was found in the first investigated region of $\mu(\cdot)$, where the construction of $\underline{V}(\cdot)$ and $\overline{V}(\cdot)$ took 12.6 seconds, and by examining 34 non-empty intersections in Algorithm 9. All results of the verification are compactly reported in Table 6.3.



(a) State profiles under $\mu(\cdot)$.

(b) State profiles under $\mu_{\mathrm{q}}(\cdot)$.

(c) Input profiles under $\mu(\cdot)$.

(d) Input profiles under $\mu_{\mathrm{q}}(\cdot)$.

Figure 6.8: Simulation results for the example inverted pendulum on a cart. Left column shows state and input profiles when the system is controlled by the real-valued MPC feedback. The column on the right corresponds to profiles obtained by controlling the system by the quantized feedback. The initial condition was $p = 0.4$, $\dot{p} = \phi = \dot{\phi} = 0$.

6.2 Safety Verification of Implicitly Defined MPC Feedback Laws

It is well known that Model Predictive Control (MPC) feedback strategies can provide an optimal operation of the plant while taking constraints into account (cf. Chapter 3). However, certain safety specifications such as performance constraints (e.g., limits on overshoots and settling time) or obstacle avoidance constraints are difficult to impose in the standard context of convex MPC because they lead to

	Operation	Time [s]	Complexity	Result
Quantization	$\mu(x)$	16.2	187 regions	
	$\mu_{\mathrm{q}}(x)$	4.0	557 regions	
Feasibility	$\overline{V}/\underline{V}$	39.5	36 regions	*Failed*
	Certification	0.5	29 LPs	
Stability	$\overline{V}/\underline{V}$	761.0	714 regions	*Failed*
	Certification	0.06	3 LPs	
Performance	$\overline{V}/\underline{V}$	12.6	8 regions (for $\mathcal{R}_1$)	*Failed*
	Certification	0.4	34 LPs	

Table 6.3: Results of the verification for inverted pendulum on a cart.

non-convex formulations which are computationally expensive to implement in real time. If satisfaction of such safety bounds can be verified off-line, then the controller can be much simpler.

Given a model of the controlled plant $x(t+1) = f(x(t), u(t))$ and the MPC feedback strategy $u(t) = \kappa(x(t))$, the objective of this section is to provide a rigorous certificate that the closed-loop system $f(x(t), \kappa(x(t)))$ is safe in the following sense: Given a set of initial conditions $\mathcal{I}$ and a set of unsafe states $\mathcal{Z}$, determine whether there exists an initial condition $x(0) \in \mathcal{I}$ such that the MPC controller forces the closed-loop states to enter $\mathcal{Z}$. If such an initial condition exists, the controller is not safe as it eventually forces the closed-loop system to violate design specifications. On the other hand, if no such $x(0) \in \mathcal{I}$ exists, the controller is deemed safe since the set of unsafe states will never be entered by the closed-loop system.

Such a safety verification task can be tackled mainly in two ways. The first set of approaches is based on so-called barrier certificates (Prajna and Jadbabaie, 2004; Prajna et al., 2007; Wieland and Allgöwer, 2007), which are closely related to the concept of Lyapunov functions used for stability analysis. The downside of such approaches is that construction of the barrier certificates is usually achieved via convex relaxations to obtain a computationally tractable problem, and is thus conservative. Therefore such approaches might fail at finding the desired safety certificate even if one exists.

The second set of methods is based on reachability analysis where one investigates whether the set $\mathcal{Z}$ is *reachable* by the closed-loop system from a given set of

initial conditions $\mathcal{I}$ (Asarin et al., 2002; Bemporad et al., 2000a; Henzinger et al., 1997; Silva et al., 2000; Stursberg and Krogh, 2003). The reachability analysis is typically performed by computing forward reachable sets, followed by determining whether the intersection between such reachable sets and the set of unsafe states is empty or not. The reachability-based procedure has several limitations, though. First, and most importantly, it assumes that the analytic description of the closed-loop dynamics is known. MPC strategies, however, only describe the optimal control inputs *implicitly* as the optimal solution to an optimal control problem. Hence the analytic form of the closed-loop dynamics is not directly available. One way around this issue is to derive the *explicit* solution of MPC (Bemporad et al., 2000b, 2002b) by employing parametric optimization (Borrelli, 2003b; Gal and Nedoma, 1972; Willner, 1967). Such solutions, however, are often very complex and difficult to construct especially for problems of large dimensionality.

In this work we take a different route which allows to investigate closed-loop systems without the need to compute the underlying explicit solution. Specifically, we show how to represent the closed-loop evolution of the feedback system by the KKT conditions (Boyd and Vandenberghe, 2004) of the MPC optimization problem. However, even for MPC problems based on linear prediction models and with all constraints being linear, the KKT conditions are nonlinear. Therefore we show how to convert such nonlinearities, in a non-conservative manner, to linear inequalities which involve continuous and binary decision variables. This allows us to provide a non-conservative answer to the safety verification problem.

The second limitation of reachability-based approaches to safety verification is that they require computing either exact or approximate reachable sets (Torrisi, 2003). Computation of exact reachable sets is expensive in large dimensions. Approximate sets (Stursberg and Krogh, 2003) are easier to construct, but may lead to conservative safety certificates. In our proposed method the construction of reachable sets is avoided altogether. Instead, the safety verification problem is posed as a series of MILP of tractable size.

Finally, the common drawback of reachability-based approaches is that they only provide a certificate of safety for a finite number of time steps. In this section we show that under mild assumptions on the terminal set included in the MPC problem, safety can be verified *ad infinitum*, i.e., for an infinite number of time steps.

6.2.1 Problem Statement

Let us consider the same setup as in Section 6.1.1, i.e. the discrete-time LTI systems of the form (6.1), subjected to (6.2). The constrained finite-time optimal control problem for the prediction model in (6.1) is given by

$$U_{\text{ol}}^{\star} = \arg\min\ x_N^T Q_N x_N + \sum_{k=0}^{N-1} x_k^T Q_x x_k + u_k^T Q_u u_k \tag{6.43a}$$

$$\text{s.t. } x_{k+1} = Ax_k + Bu_k,\ k = 0, \ldots, N-1, \tag{6.43b}$$

$$x_k \in \mathcal{X},\ k = 0, \ldots, N-1, \tag{6.43c}$$

$$u_k \in \mathcal{U},\ k = 0, \ldots, N-1, \tag{6.43d}$$

$$x_N \in \mathcal{X}_f, \tag{6.43e}$$

where x_k and u_k are, respectively, predictions of states and inputs at the k-th step of the prediction horizon (denoted by N), initialized by x_0, the measurement (or estimate) of the current state. Moreover, $Q_N = Q_N^T \succeq 0$, $Q_x = Q_x^T \succeq 0$ and $Q_u = Q_u^T \succ 0$ denote weighting matrices, and $\mathcal{X}_f \subseteq \mathcal{X}$ is a polyhedral terminal set. Finally, $U_{\text{ol}}^{\star}$ denotes the open-loop sequence of optimal control moves, i.e., $U_{\text{ol}}^{\star} = [u_0^{\star T}, \ldots, {u_{N-1}^{\star}}^T]^T$, obtained by solving (6.43) for a particular initial condition x_0.

The receding-horizon implementation of the MPC feedback law is obtained by calculating the open-loop sequence $U_{\text{ol}}^{\star}$ for a particular initial condition $x_0 = x(t)$ at each sampling step, but only employing its first element, i.e., $u_0^{\star}$, as the closed-loop control action. Hence, the RHC feedback law $\kappa : \mathbb{R}^n \to \mathbb{R}^m$ is given by

$$\kappa(x(t)) = \underbrace{\begin{bmatrix} \mathbf{I}_{m\times m} & \mathbf{0}_{m\times m} & \cdots & \mathbf{0}_{m\times m} \end{bmatrix}}_{\Phi} U_{\text{ol}}^{\star}(x(t)). \tag{6.44}$$

Since $U_{\text{ol}}^{\star}(x(t))$ in (6.44) is implicitly defined as the solution of the numerical optimization problem (6.43), we refer to (6.44) as the *implicitly defined MPC feedback law*.

The closed-loop evolution of (6.1) subject to the MPC feedback in (6.44) is defined by

$$x_{\text{cl}}(t+1) = Ax_{\text{cl}}(t) + B\kappa(x_{\text{cl}}(t)). \tag{6.45}$$

When initialized from $x(0)$, , the closed-loop state at any $t > 0$ is

$$x_{\text{cl}}(t) = A^t x(0) + \sum_{i=0}^{t-1} A^{t-i-1} B\kappa(x(i)). \tag{6.46}$$

The problem we aim at solving is to verify whether the parameters of (6.43) have been chosen such that the implicitly defined feedback law (6.44) forces the closed-loop state trajectory (6.46) *to avoid* a known set of unsafe states. If a trajectory entering the unsafe set exists, the controller is poorly designed as it does not exhibit required safety properties. Moreover, existence of such a unsafe closed-loop trajectory serves as a certificate of lack of safety. If no such trajectory entering the unsafe set exists, the controller is deemed safe. Such problem is illustrated in Figure 6.9.

We distinguish between two versions of such a problem. One verifies whether the set of unsafe states can be reached from a given set of initial conditions in finite time:

Problem 6.2.1 (Finite-time safety verification) *Let the LTI system* (6.1)*, the implicitly defined MPC feedback law* (6.44)*, the set of investigated initial conditions* $\mathcal{I} \subseteq \mathbb{R}^n$ *and the set of unsafe states* $\mathcal{Z} \subseteq \mathbb{R}^n$ *be given. Moreover, let the integer* $P < \infty$ *be given. Provide a certificate that* $x_{cl}(t) \notin \mathcal{Z}$ *for all* $t \leq P$ *with* $x_{cl}(t)$ *as in* (6.46)*.*

Note that Problem 6.2.1 can only certify safety of the MPC controller up to $t = P$, but not for $t > P$. Therefore, the second more general and more useful version certifies safety *ad infinitum*, i.e., that the set of unsafe states cannot be reached in a possibly infinite number of time steps:

Problem 6.2.2 (Infinite-time safety verification) *With the same inputs as in Problem 6.2.1, provide a certificate that* $x_{cl}(t) \notin \mathcal{Z}$ *for all* $t \leq \infty$*.*

Two choices of the set of initial conditions $\mathcal{I}$ are typically considered. One option is to choose $\mathcal{I} = \mathrm{dom}(\kappa)$ where $\mathrm{dom}(\kappa)$ is the feasibility set of (6.43). In such a settings we verify the safety and properties for all feasible initial conditions. Alternatively, $\mathcal{I} \subseteq \mathrm{dom}(\kappa)$, in which case only a subset of the feasible initial conditions is investigated (for instance the typical process operating conditions).

Remark 6.2.3 *The MPC problem* (6.43) *could be formulated to directly include the safety constraint* $x_k \notin \mathcal{Z}$ *by imposing* $x_k \in \mathcal{X} \setminus \mathcal{Z}$ *where "*$\setminus$*" is the set difference operator. However, such constraints are, in general, non-convex and would make the MPC computationally impossible to solve, especially in real time. Moreover, unless additional conditions are also employed, the avoidance of the set of unsafe*

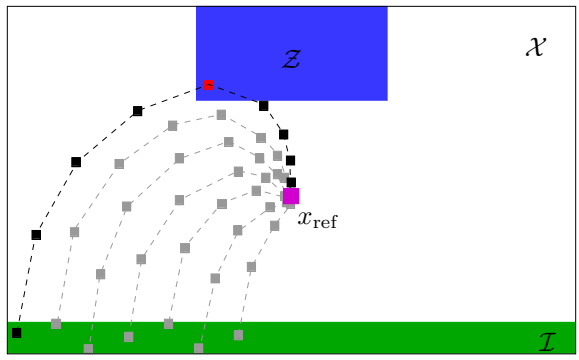


Figure 6.9: Illustrative example of the safety verification. Black borders of the picture represent the feasible domain $\mathcal{X}$ of MPC. The set of all unsafe states $\mathcal{Z}$ is denoted by blue color, green set is the set of all possible initial conditions $\mathcal{I}$, pink square is the reference and squared-dashed lines represent randomly chosen samples of the closed-loop state profiles. The state that violations safety property is denoted by the red square and the corresponding trajectory is emphasized by black color.

states would not be guaranteed ad infinitum. *Finally, in this work we only aim at verifying whether* $x_{cl}(t) \notin \mathcal{Z}$ *for a specific range of initial conditions* $\mathcal{I}$*, not for any* $x(0) \in \mathcal{X}$.

6.2.2 Safety Verification

In this section we propose a non-conservative procedure for solving Problems 6.2.1 and 6.2.2. The presented technical solution is based on the following assumptions:

Assumption 6.2.4 *The set of initial conditions* $\mathcal{I}$ *contains at least one point* $x_0 \in \mathcal{I}$ *which is a feasible initial condition for* (6.43).

Assumption 6.2.5 *The set of unsafe states* $\mathcal{Z}$ *is a convex polyhedron represented by* $\mathcal{Z} = \{x \in \mathbb{R}^n \,|\, Sx \leq s\}$. *Moreover,* $\mathcal{I}$ *is also a polyhedron.*

Assumption 6.2.6 *The set of unsafe states* $\mathcal{Z}$ *does not intersect the set of initial conditions* $\mathcal{I}$*, i.e.,* $\mathcal{Z} \cap \mathcal{I} = \emptyset$.

Assumption 6.2.4 is non-restrictive and merely requires the user to choose the initial set which is consistent with constraints of the MPC problem (6.43). Assumption 6.2.5 is required to obtain a computationally tractable and non-conservative solution to the safety verification problems. Finally, Assumption 6.2.6 is quite natural and not restrictive as it merely excludes inconsistent scenarios where the MPC problem (6.43) is set up in such a way that it forces violation of safety bounds by starting from the unsafe set directly.

We start by converting the optimal control problem (6.43) into a quadratic program. With the substitution

$$x_k = A^k x_0 + \sum_{i=0}^{k-1} A^{k-i-1} B u_i, \tag{6.47}$$

the *open-loop* profile of predicted states in (6.43), i.e., $X_{\text{ol}} = [{x_0}^T, \dots, {x_N}^T]^T$ can be compactly written as

$$X_{\text{ol}} = \Gamma x_0 + \Psi U_{\text{ol}}, \tag{6.48}$$

with

$$\Gamma = \begin{bmatrix} \mathbf{I} \\ A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}, \Psi = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ B & \mathbf{0} & \dots & \mathbf{0} \\ AB & B & \ddots & \vdots \\ \vdots & \vdots & \ddots & \mathbf{0} \\ A^{N-1}B & A^{N-2}B & \dots & B \end{bmatrix}. \tag{6.49}$$

With the substitution (6.48), the open-loop optimal control problem (6.43) can be rewritten, after straightforward algebraic manipulations (see Section 3.2.6), into

$$U_{\text{ol}}^\star(x_0) = \arg\min_{U_{\text{ol}}} \ {}^1\!/_2 U_{\text{ol}}^T H U_{\text{ol}} + x_0^T F U_{\text{ol}} \tag{6.50a}$$

$$\text{s.t. } G U_{\text{ol}} \leq w + E x_0, \tag{6.50b}$$

which is a strictly convex parametric quadratic program due to the assumption that $Q_u \succ 0$, $Q_N \succeq 0$, and $Q_x \succeq 0$. Moreover, we define

$$X_{\text{cl}}^P = [x_{\text{cl}}(0)^T, \dots, x_{\text{cl}}(P)^T]^T \tag{6.51}$$

as the *closed-loop* trajectory of (6.1) subject to the MPC feedback law (6.44) over P discrete time steps, with $x_{\text{cl}}(0) = x(0)$, and $x_{\text{cl}}(j)$ is given by (6.46) for each $j = 1, \dots, P$.

In what follows we show how to solve Problems 6.2.1 and 6.2.2 based on the assumption that the open-loop profile $X_{\rm ol}$ from (6.48) is equal to the closed-loop trajectory $X_{\rm cl}^N$ from (6.51), and $N = P$ in Problem 6.2.1.

Remark 6.2.7 *Conditions under which $X_{ol} = X_{cl}^N$ are elaborated in Mayne et al. (2000). One such a condition is that the terminal penalty Q_N, the terminal set $\mathcal{X}_f$, and the prediction horizon N are chosen such that the value function in* (6.43a) *is equal to the infinite-horizon value function. As shown, e.g., in (Grieder et al., 2005), such a condition is satisfied if Q_N is the solution to the discrete-time algebraic Riccati equation, $\mathcal{X}_f$ is the positively invariant set where the LQR controller satisfies the constraints, and the prediction horizon is sufficiently large.*

Scenario 1: Open-Loop Equals Closed-Loop

Assume that Q_N, $\mathcal{X}_f$ and N have been chosen such that the equivalence between $X_{\rm ol}$ and $X_{\rm cl}^N$ is established per Remark 6.2.7. Since $X_{\rm ol} = X_{\rm cl}^N$ is assumed, the closed-loop state profile $X_{\rm cl}^N$ in (6.51) is equal to $X_{\rm ol}$ from (6.48) where the optimal open-loop sequence of control inputs, i.e., $U_{\rm ol}^\star$, is employed. Introduce $\mathcal{M}_j \in \mathbb{R}^{n\times Nn}$ as

$$\mathcal{M}_j = \begin{bmatrix} \mathbf{0}_{n\times(j-1)n} & \mathbf{I}_{n\times n} & \mathbf{0}_{n\times(N-j)n} \end{bmatrix}. \tag{6.52}$$

Then $\mathcal{M}_j X_{\rm cl}^N = x(j)$. In other words, the matrix $\mathcal{M}_j$ extracts from the closed-loop state trajectory in (6.51) its j-th element. Reachability/unreachability of the set of unsafe states $\mathcal{Z}$ in exactly j steps with $0 \le j \le N$ can then be stated as

$$\begin{aligned}
&\text{find } x(0) && \text{(6.53a)}\\
&\text{s.t. } x(0) \in \mathcal{I}, && \text{(6.53b)}\\
&\quad \mathcal{M}_j(\Gamma x(0) + \Psi U_{\rm ol}^\star) \in \mathcal{Z}, && \text{(6.53c)}\\
&\quad U_{\rm ol}^\star \;=\; \arg\min_{U_{\rm ol}} \tfrac{1}{2} U_{\rm ol}^T H U_{\rm ol} + x(0)^T F U_{\rm ol} && \text{(6.53d)}\\
&\qquad\qquad G U_{\rm ol} \le w + E x(0), && \text{(6.53e)}\\
&\qquad\qquad x(0) \in \mathcal{I}, && \text{(6.53f)}
\end{aligned}$$

where (6.53c) translates to $x(j) \in \mathcal{Z}$ via (6.48) and (6.52).

Problem (6.53) is a *bilevel* optimization problem where $U_{\rm ol}^\star$ in (6.53c) is the optimal solution of the lower-level problem (6.53d)−(6.53f). This lower-level optimization problem implicitly defines the open-loop sequence of control inputs which

are optimal for a particular value of the initial condition $x(0)$, which is investigated in the higher-level problem. Note that the higher-level problem, represented by (6.53a)−(6.53c), is related to the lower-level problem via $x(0)$. Therefore as $x(0)$ changes in the higher-level problem, a different $U_{\text{ol}}^{\star}$ will be generated by the lower-level problem and vice versa.

Our first two results provide conditions under which a positive or a negative answer to Problem 6.2.1 exists.

Theorem 6.2.8 *Let the sets $\mathcal{I}$ and $\mathcal{Z}$ satisfy Assumptions 6.2.4 and 6.2.6, respectively. If the bilevel problem* (6.53) *is feasible for some $j \in [1, \dots, N]$ then there exists $x(0) \in \mathcal{I}$ such that $x_{cl}(j) \in \mathcal{Z}$ for some $0 < j \leq N$ (i.e., the set $\mathcal{Z}$ is reachable from $\mathcal{I}$ in, at most, N steps).*

Proof. First note that constraints (6.53b) and (6.53d)−(6.53f) can always be satisfied by a suitable choice of $x(0)$ due to Assumption 6.2.4. Therefore feasibility of (6.53) depends only on feasibility of (6.53c). Since $X_{\text{ol}} = X_{\text{cl}}^N$ is assumed, the j-th open-loop predicted state x_j is equal to the actual closed-loop state $x_{\text{cl}}(j)$. Thus (6.53c) translates to $x_{\text{cl}}(j) \in \mathcal{Z}$ due to (6.52). Therefore if (6.53) is feasible for some value of j, we have that $x(0) \in \mathcal{I}$ and $x_{\text{cl}}(j) \in \mathcal{Z}$, which shows that feasibility of (6.53) implies reachability of $\mathcal{Z}$ from $\mathcal{I}$.

Theorem 6.2.8 provides a way for finding the counter-example for the safety properties investigated in Problem 6.2.1. Such a counter-example is represented by the existence of the initial condition $x(0)$, along with the number of time steps j the closed-loop system takes to reach the set of unsafe states.

The following result is a direct corollary of Theorem 6.2.8 and establishes conditions under which $\mathcal{Z}$ cannot be reached from $\mathcal{I}$ in, at least, N steps, and thus provides a positive certificate of controller's safety according to Problem 6.2.1:

Corollary 6.2.9 *If the bilevel problems* (6.53) *is infeasible for $j = 1, \dots, N$, then there* does not *exist any $x(0) \in \mathcal{I}$ for which $x_{cl}(t) \in \mathcal{Z}$ for some $t \leq N$, i.e., the set $\mathcal{Z}$ is* not *reachable from $\mathcal{I}$ in, at least, N steps, thus $x_{cl}(t) \notin \mathcal{Z}$ for $t \leq N$.*

Next, we show that the infinite-time safety verification task of Problem 6.2.2 can be answered in finite time providing the following assumption hold:

Assumption 6.2.10 *The terminal set $\mathcal{X}_f$ in* (6.43e) *is a positively invariant set with $\mathcal{Z} \cap \mathcal{X}_f = \emptyset$.*

Existence of a positive invariant terminal set is a standard assumption in MPC to obtain closed-loop stability guarantees (see Section 3.3.2). Here, in addition we required that the terminal set is chosen not to intersect the unsafe set, i.e., the terminal set is guaranteed to be safe.

Theorem 6.2.11 *Let the sets $\mathcal{I}$ and $\mathcal{Z}$ satisfy Assumptions 6.2.4 and 6.2.6, and let $\mathcal{X}_f$ in* (6.43e) *fullfil the conditions of Assumption 6.2.10. If the bilevel problems* (6.53) *are infeasible for each $j = 1, \dots, N$, then the set $\mathcal{Z}$ is* not *reachable from $\mathcal{I}$ in any number (including infinity) of steps, i.e., $x_{cl}(t) \notin \mathcal{Z}$ for all $t > 0$.*

Proof. If (6.53) is infeasible for all $j = 1, \dots, N$, then either $\mathcal{Z}$ is unreachable in at least N steps, or it could be reached with more than N steps. The former case is already covered by Corollary 6.2.9 and thus $x_{\text{cl}}(t) \notin \mathcal{Z}$ for $t = 1, \dots, N$. The latter case is impossible under Assumption 6.2.10. To see this, note that positive invariance of $\mathcal{X}_f$ means that $x_{\text{cl}}(t+k) \in \mathcal{X}_f$ for any $k > 0$ once $x_{\text{cl}}(t) \in \mathcal{X}_f$. Since $X_{\text{ol}} = X_{\text{cl}}^N$ is assumed, x_N (which is equal to $x_{\text{cl}}(N)$) will be contained in the terminal set $\mathcal{X}_f$ via (6.43e). Thus from positive invariance of the terminal set we have $x_{\text{cl}}(N+k) \in \mathcal{X}_f$ for any $k > 0$. Finally, since $\mathcal{Z} \cap \mathcal{X}_f = \emptyset$ by Assumption 6.2.10, we have that $x_{\text{cl}}(N+k) \notin \mathcal{Z}$ for all $k > 0$. Therefore $x_{\text{cl}}(t) \notin \mathcal{Z}$ for all $t > 0$.

The safety verification taks of Problems 6.2.1 and 6.2.2 for the scenario discussed here can thus be solved by determining the feasibility of the bilevel optimization problem (6.53) for $j = 1, \dots, N$, where N is the prediction horizon in (6.43). If the problem is feasible for a particular j, the answer to Problems 6.2.1 and 6.2.2 is that the set $\mathcal{Z}$ of unsafe states is reachable from some $x(0) \in \mathcal{I}$, and the controller is thus not safe. In such a case further values of j need not be considered. Moreover, a feasible solution to the bilevel optimization problem also provides the initial condition which serves as a counter-example to safety verification.

On the other hand, if the bilevel problem (6.53) are infeasible for all $j = 1, \dots, N$, then the answer to Problem 6.2.1 is that $\mathcal{Z}$ is not reachable, and the controller is thus safe for at least N steps. Answering Problem 6.2.2 requires that the terminal set satisfies Assumption 6.2.10. In such a case the infinite-dimensional problem reduces to a finite-dimensional one.

In the sequel we show how to determine feasibility of (6.53) in a non-conservative fashion by converting it to a mixed-integer linear program. To do so, we first

formulate the KKT conditions of the lower-level problem in (6.53d)−(6.53f):

$$HU_{\text{ol}}^{\star} + F^T x(0) + G^T \lambda^{\star} = 0, \tag{6.54a}$$

$$GU_{\text{ol}}^{\star} \leq w + Ex(0), \tag{6.54b}$$

$$\lambda^{\star} \geq 0, \tag{6.54c}$$

$$\lambda_k^{\star}(G_k U_{\text{ol}}^{\star} - w_k - E_k x(0)) = 0, \tag{6.54d}$$

where (6.54a) is the stationarity condition, (6.54b) represents primal feasibility, (6.54c) is the dual feasibility, and (6.54d) stands for the complementary slackness condition, which is imposed for $k = 1, \ldots, n_c$, where n_c is the number of rows of G. Moreover, G_k, w_k, E_k denote the k-th row of the corresponding matrix. Since the lower-level problem is a strictly convex parametric QP, the KKT conditions (6.54) are necessary and sufficient (Boyd and Vandenberghe, 2004). However, they are nonlinear due to product between the Lagrange multipliers $\lambda^{\star}$ and the decision variables $U_{\text{ol}}^{\star}$ in (6.54d).

Such a nonlinearity can be worked around by realizing that for (6.54d) to hold, either $\lambda_k^{\star} = 0$ or $G_k U_{\text{ol}}^{\star} - w_k - E_k x(0) = 0$ for all $k = 1, \ldots, n_c$. One can introduce binary indicators $\delta_k \in \{0, 1\}$ and $\gamma_k \in \{0, 1\}$ such that

$$(\delta_k = 1) \;\Leftrightarrow\; (\lambda_k^{\star} = 0) \tag{6.55a}$$

$$(\gamma_k = 1) \;\Leftrightarrow\; (G_k U_{\text{ol}}^{\star} - w_k - E_k x(0) = 0). \tag{6.55b}$$

By applying standard rules of propositional logic Williams (1993), also known as the *big-M* technique, the equivalences in (6.55) can be furthermore rewritten into a set of inequalities that are linear in the decision variables $\lambda_k^{\star}$, $U_{\text{ol}}^{\star}$, δ_k, and γ_k,

$$-Z(1 - \delta_k) \leq \lambda_k^{\star} \leq Z(1 - \delta_k), \tag{6.56a}$$

$$-Z(1 - \gamma_k) \leq G_k U_{\text{ol}}^{\star} - w_k - E_k x(0) \leq Z(1 - \gamma_k), \tag{6.56b}$$

where Z is a sufficiently large constant. It is trivial to verify that if $\delta_k = 1$ in (6.56a), then $\lambda_k^{\star} = 0$ is the only feasible value. If $\delta_k = 0$, then (6.56a) is inactive. Similar reasoning holds for (6.56b). Then the complementarity slackness condition (6.54d) can be equivalently written as the propositional logic statement of the form $\delta_k \vee \gamma_k$ (i.e., either the k-th Lagrange multiplier is zero, or the k-th constraint is active), or, equivalently, be written as $\delta_k + \gamma_k \geq 1$. Therefore the KKT conditions (6.54)

can be equivalently written as

$$HU_{\text{ol}}^\star + F^T x(0) + G^T \lambda^\star = 0, \tag{6.57a}$$
$$GU_{\text{ol}}^\star \leq w + Ex(0), \tag{6.57b}$$
$$\lambda^\star \geq 0, \tag{6.57c}$$
$$-Z(1-\delta_k) \leq \lambda_k^\star \leq Z(1-\delta_k), \tag{6.57d}$$
$$-Z(1-\gamma_k) \leq G_k U_{\text{ol}}^\star - w_k - E_k x(0) \leq Z(1-\gamma_k), \tag{6.57e}$$
$$\delta_k + \gamma_k \geq 1, \tag{6.57f}$$

where (6.57d)−(6.57f) are imposed for $k = 1, \ldots, n_c$.

In what follows we will abbreviate (6.57) by $\text{KKT}(x(0), U_{\text{ol}}^\star, \lambda^\star, \delta, \gamma) \leq 0$. Accordingly, the bilevel optimization problem (6.53) can be equivalently written as

$$\text{find } x(0) \tag{6.58a}$$
$$\text{s.t. } x(0) \in \mathcal{I}, \tag{6.58b}$$
$$S(\mathcal{M}_j(\Gamma x(0) + \Psi U_{\text{ol}}^\star)) \leq s, \tag{6.58c}$$
$$\text{KKT}(x(0), U_{\text{ol}}^\star, \lambda^\star, \delta, \gamma) \leq 0, \tag{6.58d}$$

where (6.58c) is equivalent to (6.53c) and $\mathcal{Z}$ is assumed to be a polyhedron, cf. Assumption 6.2.5. Since all constraints are linear (cf., (6.57)), problem (6.58) is a mixed-integer feasibility problem with continuous decision variables $x(0) \in \mathbb{R}^n$, $U_{\text{ol}}^\star \in \mathbb{R}^{Nm}$, $\lambda^\star \in \mathbb{R}^{n_c}$, and binary decision variables $\delta \in \{0,1\}^{n_c}$ and $\gamma \in \{0,1\}^{n_c}$, where n_c is the number of constraints of the pQP formulation of the MPC problem in (6.50).

Remark 6.2.12 *Note that showing safety of the MPC feedback per Corollary 6.2.9 and Theorem 6.2.11 relies on infeasibility of* (6.58) *for each* $j \in [1, \ldots, N]$. *To prevent numerical problems which might lead to false indication of infeasibility, we propose to soften the hard constraints* (6.58c) *by*

$$S(\mathcal{M}_j(\Gamma x(0) + \Psi U_{ol}^\star)) \leq s + \omega, \tag{6.59a}$$
$$\omega \geq 0, \tag{6.59b}$$

where $\omega \in \mathbb{R}^{n_S}$ *are the slack variables (here,* n_S *is the number of rows of* S *in Assumption 6.2.5). Moreover, the objective* (6.58a) *should be replaced by* $\min \|\omega\|_1$.

Such a modified problem is always feasible. If $\omega = 0$ *in the modified problem,* (6.58) *is feasible by Assumption 6.2.4. If* $\omega_i > 0$ *for at least one component of* ω *in the modified problem, then* (6.58) *is infeasible.*

Finally, we remark that even though the MILP problem (6.58) is non-convex due to presence of binary decision variables, its feasibility can always be determined in finite time, and the optimal solution of the modified problem per Remark 6.2.12 can always be found in finite time, e.g. by branch-and-bound methods.

Scenario 2: Open-Loop not Equal to Closed-Loop

If a mismatch between the predicted open-loop state profile X_{ol} and the actual closed-loop response X_{cl}^P is assumed, the closed-loop control action at the j-th step, i.e., $u(j)^\star = \kappa(x_{\text{cl}}(j))$ is *not* necessarily equal to the j-th element of $U_{\text{ol}}^\star(x(0))$. Therefore $x(1) = Ax(0) + B\kappa(x(0))$, where $\kappa(x(0)) = \Phi U_{\text{ol}}^\star(x(0))$, Φ is as in (6.44), and $U_{\text{ol}}^\star(x(0))$ is the open-loop optimal control sequence corresponding to the initial condition $x_0 = x(0)$ in (6.43). To simplify the notation, we will abbreviate $U_{\text{ol}}^\star(x(0))$ by $U_{\text{ol}}^\star(0)$. As elaborated in Section 6.2.2, $U_{\text{ol}}^\star(0)$ will be the optimal open-loop sequence for the initial condition $x(0)$ if and only if there exist associated Lagrange multipliers $\lambda^\star(0)$, along with binary vectors $\delta(0)$ and $\gamma(0)$, such that the KKT system (6.58) holds. For $j = 2$, we have $x_{\text{cl}}(2) = Ax_{\text{cl}}(1) + B\kappa(x_{\text{cl}}(1))$, where $\kappa(x_{\text{cl}}(1)) = \Phi U_{\text{ol}}^\star(x_{\text{cl}}(1)) = \Phi U_{\text{ol}}^\star(1)$. The corresponding optimizer $U_{\text{ol}}^\star(1)$ is, again, implicitly given as the feasible solution to the KKT system (6.58) for $x_0 = x_{\text{cl}}(1)$, i.e., $U_{\text{ol}}^\star(1)$ is optimal if and only if $\text{KKT}(x_{\text{cl}}(1), U_{\text{ol}}^\star(1), \lambda^\star(1), \delta(1), \gamma(1)) \leq 0$ holds. The same argument applies for $j = 3, \ldots, P$.

Therefore, to determine whether $\mathcal{Z}$ is reachable from $\mathcal{I}$, we need to determine existence of a whole set of open-loop sequences $\{U_{\text{ol}}^\star(0), U_{\text{ol}}^\star(1), \ldots, U_{\text{ol}}^\star(j-1)\}$:

$$\begin{aligned}
&\text{find } x(0) && \text{(6.60a)}\\
&\text{s.t. } x(0) \in \mathcal{I}, && \text{(6.60b)}\\
&\quad x_{\text{cl}}(j) \in \mathcal{Z}, && \text{(6.60c)}\\
&\quad x_{\text{cl}}(p+1) = Ax_{\text{cl}}(p) + B\Phi U_{\text{ol}}^\star(p), && \text{(6.60d)}\\
&\quad \text{KKT}(x_{\text{cl}}(p), U_{\text{ol}}^\star(p), \lambda^\star(p), \delta(p), \gamma(p)) \leq 0, && \text{(6.60e)}
\end{aligned}$$

where constraints (6.60d) and (6.60e) are imposed for $p = 0, \ldots, j-1$ together with the substitution $x(0) = x_{\text{cl}}(0)$ to enable the recursion in (6.60d). Since $\mathcal{I}$

and $\mathcal{Z}$ are assumed to be polyhedra, and because (6.60e) can be cast as a set of mixed-integer inequalities per (6.57), problem (6.60) for a finite j is a mixed-integer feasibility problem in decision variables $x(0), x_{\rm cl}(1), \ldots, x_{\rm cl}(j)$ with $x_{\rm cl}(p) \in \mathbb{R}^n$, $U^\star_{\rm ol}(0), \ldots, U^\star_{\rm ol}(j-1)$ with $U^\star_{\rm ol}(p) \in \mathbb{R}^{Nm}$, $\lambda^\star(0), \ldots, \lambda^\star(j-1)$ with $\lambda^\star(p) \in \mathbb{R}^{n_c}$, and binary decision variables $\delta(0), \ldots, \delta(j-1)$ with $\delta(p) \in \{0,1\}^{n_c}$ and $\gamma(0), \ldots, \gamma(j-1)$ with $\gamma(p) \in \{0,1\}^{n_c}$, where $p = 0, \ldots, j-1$.

The following theorem is a direct extension of Theorem 6.2.8 and Corollary 6.2.9 and provides a technical solution to the finite-time safety verification task of Problem 6.2.1 for the $X_{\rm ol} \neq X_{\rm cl}$ scenario:

Theorem 6.2.13 *Let the sets $\mathcal{I}$ and $\mathcal{Z}$ satisfying Assumptions 6.2.4 and 6.2.6 be given, and let $P < \infty$. If the mixed-integer feasibility problem* (6.60) *is feasible for some $j \in [1, \ldots, P]$, then $\mathcal{Z}$ is reachable from $\mathcal{I}$ under the MPC feedback* (6.44) *in exactly j steps. If* (6.60) *is infeasible for all $j = 1, \ldots, P$, then $\mathcal{Z}$ is unreachable from $\mathcal{I}$ in, at least, P steps.*

Proof. Directly by Theorem 6.2.8 and Corollary 6.2.9.

To provide a technical solution to the infinite-time safety verification Problem 6.2.2, we require the following technical assumption:

Assumption 6.2.14 *The terminal set $\mathcal{X}_f$ in* (6.43e) *is a positively invariant set, $\mathcal{Z} \cap \mathcal{X}_f = \emptyset$, and the parameters N, Q_N, Q_x, Q_u of the MPC optimal control problem* (6.43) *are chosen such that the controller forces the state of the closed-loop system* (6.45) *to enter $\mathcal{X}_f$ in, at most, P steps.*

Theorem 6.2.15 *Let $\mathcal{I}$, $\mathcal{Z}$, and $\mathcal{X}_f$ satisfy, respectively, Assumptions 6.2.4, 6.2.6, and 6.2.14. If* (6.60) *are infeasible for each $j = 1, \ldots, P$, then $\mathcal{Z}$ is unreachable from $\mathcal{I}$ in any number of steps, i.e., $x_{cl}(t) \notin \mathcal{Z}$ for any $t > 0$.*

Proof. First, per Theorem 6.2.13, infeasibility of (6.60) for each $j = 1, \ldots, P$ implies $\mathcal{Z}$ cannot be reached in P steps, i.e., $x_{\rm cl}(t) \notin \mathcal{Z}$ for $t = 1, \ldots, P$. Second, under Assumption 6.2.14 we have that $x_{\rm cl}(P) \in \mathcal{X}_f$. Since $\mathcal{X}_f$ is positively invariant, then $x_{\rm cl}(P+p) \in \mathcal{X}_f$ for any $k > 0$. Finally, since $\mathcal{Z} \cap \mathcal{X}_f = \emptyset$, it follows that $x_{\rm cl}(P+p) \notin \mathcal{Z}$ for any $k > 0$. Combining both statements gives $x_{\rm cl}(t) \notin \mathcal{Z}$ for any $t > 0$.

Complexity of the decision problem (6.60) is larger than that of (6.58). Specifically, (6.60) has j-times the number of binary variables compared to (6.58). In the

worst case, one needs to solve (6.60) a total of P times, each time introducing a new set of decision variables $(U^{\star}_{\mathrm{ol}}(j), \lambda^{\star}(j), \delta(j), \gamma(j))$.

Remark 6.2.16 *In practice, it may be difficult to derive (or even to estimate) P directly from parameters of the open-loop MPC problem in* (6.43)*. One option is to choose a conservatively large value of P, followed by using bisection. Alternatively, an a-priori upper bound on P can be set and be interpreted as the worst acceptable liveness of the controller. Then infeasibility of* (6.60) *implies that the MPC controller either does not force the closed-loop states to enter $\mathcal{Z}$, or that $\mathcal{Z}$ could be reached, but after an unacceptably long time.*

Remark 6.2.17 *The scenario discussed here can even be extended to cover cases where the MPC controller is synthesized based on the prediction model $x(t+1) = Ax(t) + Bu(t)$, but its safety properties are verified under the assumption that the calculated control actions are applied to a different system, say $\tilde{x}(t+1) = \widetilde{A}\tilde{x}(t) + \widetilde{B}u(t)$. The extended case can be recovered by replacing A by $\widetilde{A}$ and B by $\widetilde{B}$ throughout the reported results.*

Scenario 3: Verification of a Quantized Feedback

Next we will elaborate scenario where actuators, can operate only with a finite number of control levels. In other words, assume that the control actions $u(t)$ are subjected to a quantizer that disposes of D quantization levels $q = [q_1, \ldots, q_d]^T$, $q_i \neq q_j$, $\forall i \neq j$. Let us now denote $u_q(t) \in \mathbb{R}^m$ to be a quantized counterpart to a real-valued control input $u(t) \in \mathbb{R}^m$, then the rounding-based rule $f_q : \mathbb{R}^m \to \mathbb{R}^m$ can be given by

$$u_q(t) := q_i \;\text{ if }\; u(t) \in \mathcal{P}_i, \tag{6.61}$$

where

$$\mathcal{P}_i = \{u(t) \mid ||u(t) - q_i||_2 \leq ||u(t) - q_j||_2,\ \forall j \neq i\}, \tag{6.62}$$

represents the input-space polydedral region in which the rounding-based rule in (6.61) maps each real-valued control input into the nearest quantization level q_i. Denote next $U^{\star}_{\mathrm{ol,q}}$ to be a quantized open-loop sequence of optimal control inputs, i.e. , $U^{\star}_{\mathrm{ol,q}} = [u^T_{q,1}, \ldots, u^T_{q,N-1}]^T$ and $\kappa_q(x_q(t)) = \Phi U^{\star}_{\mathrm{ol,q}}(x_q(t))$ to be the RHC feedback of (6.43) w.r.t. (6.61), where the matrix Φ is the same as in (6.44), then the closed-loop state profile of (6.1) is given by

$$x_{\mathrm{q}}(t+1) = Ax_{\mathrm{q}}(t) + B\kappa_q(x_q(t)), \tag{6.63}$$

and the state evolution $x_{\mathrm{q}}(t)$, initialized from $x(0)$, can be expressed as

$$x_{\mathrm{q}}(t) = A^t x(0) + \sum_{i=0}^{t-1} A^{t-i-1} B \kappa_q(x_q(i)). \tag{6.64}$$

Now, the objective is to provide a rigorous certificate, that will determine whether there does not exists any initial condition $x(0) \in \mathcal{I}$ for which the MPC policy (6.43) will generate control actions $u_q(t) \in \{q_1, \ldots, q_d\}$ such that $x_{\mathrm{q}}(t) \in \mathcal{Z}$, where $x_{\mathrm{q}}(t)$ is given by (6.64). In sequel we will show how to rewrite the quantization law in (6.61) into series of linear expressions that will be subsequently employed to the verification problem.

We start by devise polyhedral regions $\mathcal{P}_i \in \mathbb{R}^m$, each of which represents a domain of real-valued control inputs $u(t)$ where a particular quantization level $u_q(t) \in \{q_1, \ldots, q_d\}$, $q_i \neq q_j$, $\forall i \neq j$ is active. As we have already know from Section 6.1.2, the Voronoi diagram can be employed to determine the closest quantization level q_i for a given value of $u(t)$. Let us therefore rewrite (6.13) into a polyhedron

$$\mathcal{P}_i = \{u(t) \mid P_{A,i} u(t) \leq P_{B,i}\}, \tag{6.65}$$

where

$$P_{A,i} = \begin{bmatrix} 2(q_1 - q_i)^T \\ \vdots \\ 2(q_{i-1} - q_i)^T \\ 2(q_{i+1} - q_i)^T \\ \vdots \\ 2(q_d - q_i)^T \end{bmatrix}, P_{B,i} = \begin{bmatrix} q_1^T q_1 - q_i^T q_i \\ \vdots \\ q_{i-1}^T q_{i-1} - q_i^T q_i \\ q_{i+1}^T q_{i+1} - q_i^T q_i \\ \vdots \\ q_d^T q_d - q_i^T q_i \end{bmatrix},$$

which needs to be furthermore intersected with $\mathcal{U}$. Furthermore, from the property of Voronoi diagram, we have that $\mathrm{int}(\mathcal{P}_i) \cap \mathrm{int}(\mathcal{P}_i) = \emptyset$ and $\cup_i \mathcal{P}_i = \mathrm{dom}(\kappa_q)$ are provided. To define in which region $\mathcal{P}_i$ the current real-valued control input $u(t)$ belongs to, one can introduce binary variables $\psi \in \{0,1\}^d$ such that

$$(\psi_i = 1) \;\Leftrightarrow\; (P_{A,i} u(t) - P_{B,i} \leq 0). \tag{6.66}$$

Subsequently, inequalities in (6.66) can be rewritten by applying the *big*-M technique as

$$P_{A,i} u(t) - P_{B,i} \leq T(1 - \psi_i), \tag{6.67}$$

where $P_{A,i} \in \mathbb{R}^{d-1\times m}$, $P_{B,i} \in \mathbb{R}^{d-1}$, and $T \in \mathbb{R}^{d-1}$ is a vector of a sufficiently large scalar Z, i.e. $T = \mathbf{1}_{d-1}Z$. It can be easily shown that inequality in (6.66) holds if and only if $\psi_i = 1$, otherwise if $\psi_i = 0$ then region $\mathcal{P}_i$ is inactive. Finally, the rounding-base rule in (6.61) can by stated as

$$P_{A,i}u(t) - P_{B,i} \leq T(1-\psi_i),\ \forall i = 1,\dots,d, \tag{6.68a}$$

$$\sum_{i=1}^{d} \psi_i = 1, \tag{6.68b}$$

$$u_q(t) = \sum_{i=1}^{d} \psi_i q_i. \tag{6.68c}$$

In (6.68) we have provided a quantization formula, which maps one real-valued control input $u(t)$ into its quantized part $u_q(t)$. However, the MPC policy (6.43) optimizes the control inputs along the entire prediction horizon N. Therefore, in order to enforce a finite precision arithmetics into (6.43), i.e. $u \in q$, let us extend (6.68) for the full sequence of control moves U_{ol}. In other words, we need to repeat (6.68) N times for each predicted control input u_k, i.e.

$$u_{q,k} = q_i \quad \text{if} \quad u_k \in \mathcal{P}_i,\ k = 0,\dots,N-1,$$

to retain $U_{\text{ol,q}} = [u_{q,0}^T,\dots,u_{q,N-1}^T]^T$. Therefore, the search for the entire quantized open-loop sequence of control inputs $U_{\text{ol,q}}$ can be compactly stated as

$$P_{A,i}u_k - P_{B,i} \leq T(1-\psi_{i,k}),\ \forall i = 1,\dots,d, \tag{6.69a}$$

$$\sum_{i=1}^{d} \psi_{i,k} = 1, \tag{6.69b}$$

$$u_{q,k} = \sum_{i=1}^{d} \psi_{i,k} q_i, \tag{6.69c}$$

where (6.69a)-(6.69c) are imposed for all $k = 0,\dots,N-1$, $\psi \in \{0,1\}^{Nd}$, $P_{A,i} \in \mathbb{R}^{d-1\times m}$, $P_{B,i} \in \mathbb{R}^{d-1}$, $T = \mathbf{1}_{d-1}Z$ with sufficiently large scalar Z. For simplicity, equations in (6.69), which translate real-valued control inputs $U_{\text{ol}}(p)$ into quantized control actions $U_{\text{ol,q}}(p)$ at a given verification time step p will be compactly denoted by $f_q(U_{\text{ol}}(p), U_{\text{ol,q}}(p), \psi(p)) \leq 0$.

Finally, with the derived quantization formula (6.69) in hand, we are able to enforce MPC policy in (6.43) to generate the optimal quantized control actions $U^{\star}_{\text{ol,q}}$. By realizing that the conditions under which $X_{\text{ol}} = X_{\text{cl}}$ (cf. Remark 6.2.7)

do not hold due to the presence of the quantization effect $f_q(\cdot)$, we are forced to apply the results derived in the previous Scenario 2. Particularly, by taking (6.60) and enforcing (6.69), then the verification problem which determines whether the MPC policy (6.43) subjected to the rounding-based law (6.61) forces $x_{\rm q}(t)$ to enter $\mathcal{Z}$ from $\mathcal{I}$ can be stated as

$$\begin{aligned}
&\text{find } x(0) && \text{(6.70a)}\\
&\text{s.t. } x(0) \in \mathcal{I}, && \text{(6.70b)}\\
&\quad x_{\rm q}(j) \in \mathcal{Z}, && \text{(6.70c)}\\
&\quad x_{\rm q}(p+1) = Ax_{\rm q}(p) + B\kappa_q(x_{\rm q}(p)), && \text{(6.70d)}\\
&\quad f_q(U^\star_{\rm ol}(p), U^\star_{\rm ol,q}(p), \psi(p)) \leq 0, && \text{(6.70e)}\\
&\quad \mathrm{KKT}(x_{\rm cl}(p), U^\star_{\rm ol}(p), \lambda^\star(p), \delta(p), \gamma(p)) \leq 0, && \text{(6.70f)}
\end{aligned}$$

where constraints (6.70d)-(6.70f) are imposed for $p = 0, \ldots, j-1$ and $x(0) = x_{\rm q}(0)$ is an initialization for the recursion in (6.70d). Note that each real-valued control input $U^\star_{\rm ol}(p)$ in (6.70f) is linked with the quantization law (6.61) via (6.70e), the consequence of which is that (6.70) directly operates with $U^\star_{\rm ol,q}(p)$. Moreover, with $\mathcal{I}$ and $\mathcal{U}$ being polyhedra (see Assumption 6.2.5), and (6.70f) written as a set of mixed-integer inequalities per (6.57) and (6.69), respectively, problem (6.70) for a finite j is a mixed-integer feasibility problem in decision variables $x_{\rm q}(0), x_{\rm q}(1), \ldots, x_{\rm q}(j-1)$ with $x_{\rm q}(p) \in \mathbb{R}^n$, $U^\star_{\rm ol}(0), \ldots, U^\star_{\rm ol}(j-1)$ with $U^\star_{\rm ol}(p) \in \mathbb{R}^{Nm}$, $U^\star_{\rm ol,q}(0), \ldots, U^\star_{\rm ol,q}(j-1)$ with $U^\star_{\rm ol,q}(p) \in \mathbb{R}^{Nm}$, $\lambda(0), \ldots, \lambda(j-1)$ with $\lambda(p) \in \mathbb{R}^{n_c}$, and binary decision variables $\delta(0), \ldots, \delta(j-1)$ with $\delta(k) \in \{0,1\}^{n_c}$, $\gamma(0), \ldots, \gamma(j-1)$ with $\gamma(k) \in \{0,1\}^{n_c}$, and $\psi(0), \ldots, \psi(j-1)$ with $\psi(k) \in \{0,1\}^{Nd}$, where $k = 0, \ldots, j-1$.

In sequel, the solution for the finite-time verification Problem 6.2.1 and for the infinite-time verification task of Problem 6.2.2, when the feedback law of (6.43) respects the rounding-based quantization rule (6.61), are reported next:

Theorem 6.2.18 *Let the sets $\mathcal{I}$ and $\mathcal{Z}$ satisfying Assumptions 6.2.4 and 6.2.6 be given, and let $P < \infty$. If the mixed-integer feasibility problem* (6.70) *is feasible for some $j \in [1, \ldots, P]$, then $\mathcal{Z}$ is reachable from $\mathcal{I}$ under the MPC feedback* (6.44) *in exactly j steps. If* (6.70) *is infeasible for all $j = 1, \ldots, P$, then $\mathcal{Z}$ is unreachable from $\mathcal{I}$ in, at least, P steps.*

Proof. Follows from Theorem 6.2.8 and Corollary 6.2.9.

Assumption 6.2.19 *Let $\mathcal{X}_f$ in* (6.43e) *be a positive invariant terminal set with respect to quantization levels $q = [q_1, \ldots, q_d]^T$, i.e. $\mathcal{X}_f = \{x_q(t) \mid \exists u_q(t) \in \{q_1, \ldots, q_d\}, Ax_q(t) + Bu_q(t) \in \mathcal{X}_f, \forall t > 0\}$. Moreover, assume $\mathcal{Z} \cap \mathcal{X}_f = \emptyset$ and that the parameters N, Q_N, Q_x, Q_u of the MPC policy* (6.43) *are chosen such that the closed-loop system* (6.63) *enters $\mathcal{X}_f$ in, at most, P steps.*

Theorem 6.2.20 *Let $\mathcal{I}$, $\mathcal{Z}$, and $\mathcal{X}_f$ satisfy, respectively, Assumptions 6.2.4, 6.2.6, and 6.2.19. If* (6.70) *is infeasible for each $j = 1, \ldots, P$, then $\mathcal{Z}$ is unreachable from $\mathcal{I}$ in any number of steps, i.e., $x_q(t) \notin \mathcal{Z}$ for any $t > 0$.*

Proof. Analogical to Theorem (6.2.15).

6.2.3 Case Study: Safety Verification of MPC

Four Tanks System

We apply the procedure of Section 6.2.2 to verify safety properties of an MPC controller which governs inflows into a four tank system depicted in Figure 6.10. The dynamical behavior of such a system is represented by (Drca, 2007)

$$\dot{h}_1(t) = -\frac{k_1}{F_1}\sqrt{2gh_1(t)} + \frac{k_3}{F_3}\sqrt{2gh_3(t)} + \frac{\gamma_1}{F_1}q_a(t), \tag{6.71a}$$

$$\dot{h}_2(t) = -\frac{k_2}{F_2}\sqrt{2gh_2(t)} + \frac{k_4}{F_2}\sqrt{2gh_4(t)} + \frac{\gamma_2}{F_2}q_b(t), \tag{6.71b}$$

$$\dot{h}_3(t) = -\frac{k_3}{F_3}\sqrt{2gh_3(t)} + \frac{(1-\gamma_2)}{F_3}q_b(t), \tag{6.71c}$$

$$\dot{h}_4(t) = -\frac{k_4}{F_4}\sqrt{2gh_4(t)} + \frac{(1-\gamma_1)}{F_4}q_a(t), \tag{6.71d}$$

where $h_i(t)$, $i = 1, \ldots, 4$ are the liquid levels in corresponding tanks, $q_a(t)$ and $q_b(t)$ are the manipulated inflows, g is the gravitational constant, k_i, $i = 1, \ldots, 4$ are the valve constants, F_i, $i = 1, \ldots, 4$ represents areas of the tanks' cross-sections. Finally, γ_1 and γ_2 are constants which govern the split of inflows into the lower and upper level tanks (i.e., $q_1(t) = \gamma_1 q_a(t)$, $q_4(t) = (1-\gamma_1)q_a(t)$, $q_2(t) = \gamma_2 q_b(t)$, $q_3(t) = \gamma_2 q_b(t)$).

Linearization of (6.71) by first-order Taylor expansion (see Section 3.2.2) yields

$$\dot{x}(t) = \begin{bmatrix} -\frac{1}{T_1} & 0 & \frac{F_3}{(F_1 T_3)} & 0 \\ 0 & -\frac{1}{T_2} & 0 & \frac{F_4}{(F_2 T_4)} \\ 0 & 0 & -\frac{1}{T_3} & 0 \\ 0 & 0 & 0 & -\frac{1}{T_4} \end{bmatrix} x(t) + \begin{bmatrix} \frac{\gamma_1}{F_1} & 0 \\ 0 & \frac{\gamma_2}{F_2} \\ 0 & \frac{(1-\gamma_2)}{F_3} \\ \frac{(1-\gamma_1)}{F_4} & 0 \end{bmatrix} u(t), \tag{6.72}$$

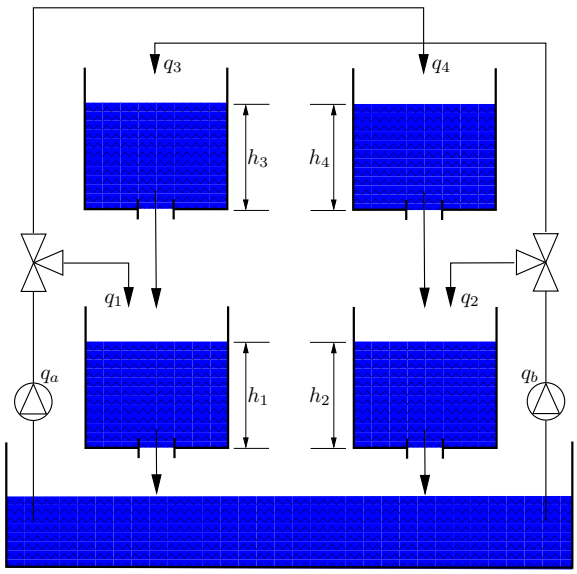


Figure 6.10: The four tanks system.

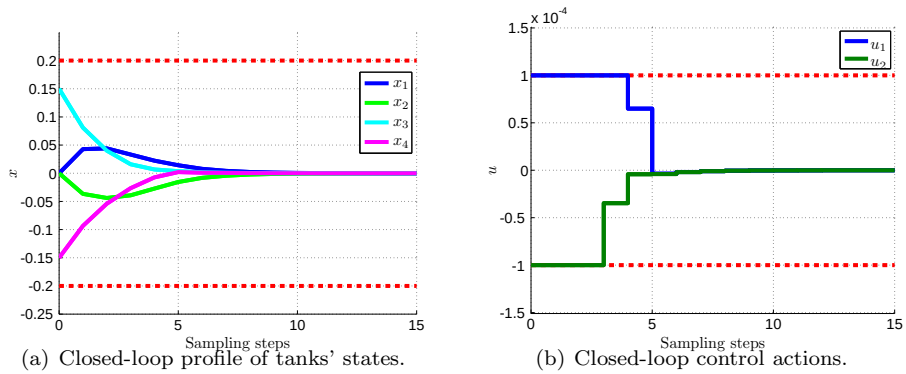
with

$$T_i = \frac{F_i}{k_i}\sqrt{\frac{2h_i^{\mathrm{s}}}{g}}, \tag{6.73}$$

where $x(t) = (h(t) - h^{\mathrm{s}})$, $u(t) = (q(t) - q^{\mathrm{s}})$ are the deviations of states and inputs from the respective steady-state values. In this case study we use $h_i^s = 0.2\,\mathrm{m}$, $i = 1, \ldots, 4$, $q_a^s = q_b^{\mathrm{s}} = 1 \cdot 10^{-4}\,\mathrm{m^3 s^{-1}}$, $F_i = 0.06\,\mathrm{m}^2$, $i = 1, \ldots, 4$, $k_1 = 8.7932 \cdot 10^{-4}$, $k_2 = 7.3772 \cdot 10^{-4}$, $k_3 = 6.3495 \cdot 10^{-4}$, $k_4 = 4.3567 \cdot 10^{-4}$, $g = 9.81\,\mathrm{ms}^{-2}$, $\gamma_1 = 0.2$, $\gamma_2 = 0.4$, and discretization of (6.72) with sampling time of 5 seconds.

Control Objective

The control objective is to manipulate the liquid levels in all four tanks to their respective steady-state values (i.e., for the deviation states $x_i(t)$ to reach zero levels), while satisfying state constraints $-0.2 \leq x_i(t) \leq 0.2$, $i = 1, \ldots, 4$ (which corresponds to $0\,\mathrm{m} \leq h_i(t) \leq 0.4\,\mathrm{m}$) and input constraints $-1 \cdot 10^{-4} \leq u_j(t) \leq 1 \cdot 10^{-4}$ (which translate to $0\,\mathrm{m^3 s^{-1}} \leq q_j(t) \leq 2 \cdot 10^{-4}\,\mathrm{m^3 s^{-1}}$). This is achieved by devising an MPC feedback strategy which solves (6.43) with $Q_x = \mathrm{diag}(1, 1, 1, 1)$, $Q_u = \mathrm{diag}(1, 1)$, Q_N equal to the solution of the algebraic Riccati equation, $\mathcal{X}_f$ being the constraint admissible set of the plant in closed-loop with the LQR con-



(a) Closed-loop profile of tanks' states.

(b) Closed-loop control actions.

Figure 6.11: Closed-loop regulation of the four tanks system from the initial condition $x_0 = [0,\ 0,\ 0.15,\ -0.15]^T$. Red dashed lines represent corresponding constraints.

troller, obtained for the same cost function of the MPC. Finally, $N = 8$. The closed-loop trajectory of the system in (6.72) subject to the MPC policy (6.44) is provided in Figure 6.11. As can be seen, the response of $x_1(t)$ and $x_2(t)$ (which represent levels in the bottom tanks) exhibits a non-minimum phase behavior, which is a consequence of $\gamma_1 < 0.5$ and $\gamma_2 < 0.5$.

Safety Verification

We wish to verify that the MPC policy provides that the overshoots and undershoots in lower tanks due to the non-minimum phase behavior do not exceed prescribed bounds. Specifically, for the set of initial conditions $\mathcal{I} = \{x(t) \mid -0.2 \leq x_{3,4}(t) \leq 0.2,\ x_{1,2}(t) = 0\}$ (i.e., bottom tanks at their steady-state levels with upper tanks being filled up to arbitrary levels within constraints) we aim at verifying that the closed-loop system avoids the sets $\mathcal{Z}_1 = \{x(t) \,|\, x_1(t) \geq 0.05\}$, $\mathcal{Z}_2 = \{x(t) \,|\, x_1(t) \leq -0.05\}$, $\mathcal{Z}_3 = \{x(t) \,|\, x_2(t) \geq 0.05\}$, and $\mathcal{Z}_4 = \{x(t) \,|\, x_2(t) \leq -0.05\}$. The reasoning behind such a choice is verifying whether the MPC controller rejects disturbances in the upper tanks without large changes of the levels in the bottom tanks.

Remark 6.2.21 *It may not be desirable to include* $-0.05 \leq x_{1,2}(t) \leq 0.05$ *as hard constraints in* (6.43) *since it would render the MPC problem infeasible for several initial conditions. The objective here is to verify whether the MPC policy is tuned*

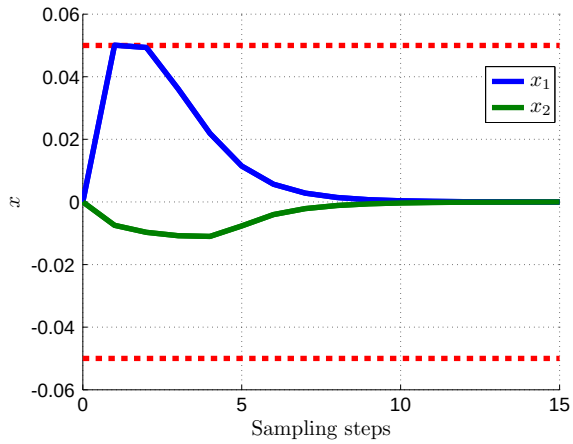


Figure 6.12: Example of an unsafe trajectory of lower tanks which starts from $x(0) = [0,\ 0,\ 0.18,\ -0.02]^T$ and violates the limits of maximal under/over-shoots, represented by the dotted black lines.

in such a way that it "voluntarily" maintains these limits for a specific *range of initial conditions.*

Since the conditions of Remark 6.2.7 are satisfied, the open-loop predicted sequence is equal to the actual closed-loop response, thus we can use the proposed procedures to solve the infinite-time verification task of Problem 6.2.2 by employing Theorems 6.2.8 and 6.2.11. Specifically, we have formulated (6.58) using YALMIP (Löfberg, 2004) and solved the resulting MILPs by CPLEX. After 0.8 seconds[7] a feasible solution to (6.58) was found which, according to Theorem 6.2.8 means that the safety specifications are violated. The associated counter-example is represented by the initial condition $x(0) = [0,\ 0,\ 0.18,\ -0.02]^T$, for which the MPC feedback forces $x_1(t)$ to exceed 0.05, as can be seen in Figure 6.12

Finding the Largest Bound for the Initial Conditions

Next, we have applied the safety verification procedure to find out the largest bound a in $\mathcal{I} = \{x(0) \mid -a \leq x_{3,4}(0) \leq a,\ x_{1,2}(0) = 0\}$ for which the safety criteria would be satisfied *ad infinitum*. To do so, we have applied bisection when solving problem (6.58) for various values of a (increasing a if safety can be shown,

[7] On a 1.7 GHz CPU running Matlab R2013a.

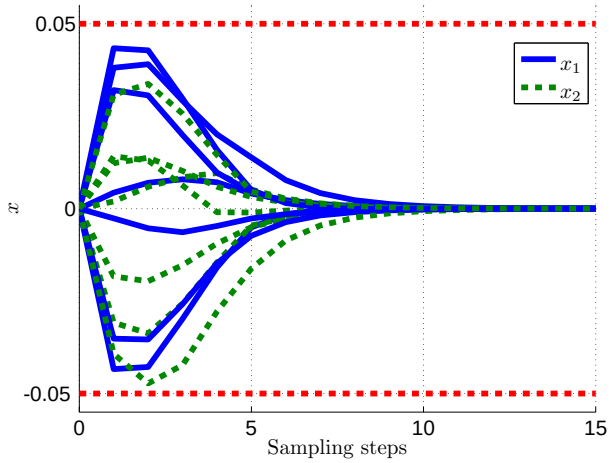


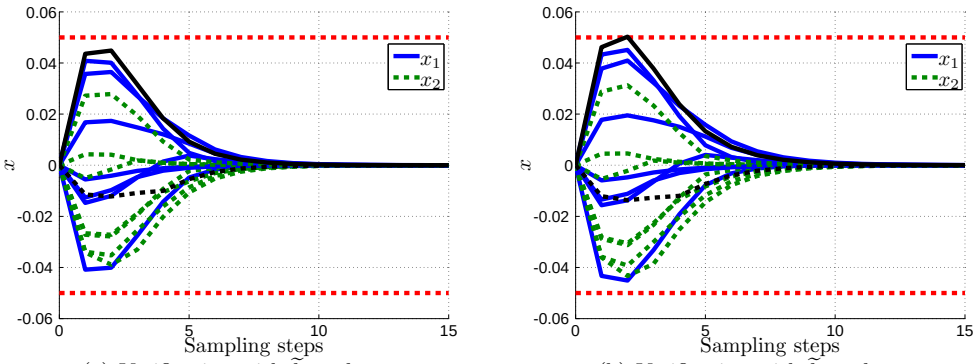
Figure 6.13: Safe closed-loop trajectories generated for randomly selected initial conditions from the set $\mathcal{I} = \{x(0) \mid -a^\star \leq x_{3,4}(0) \leq a^\star,\ x_{1,2}(0) = 0\}$ for $a^\star = 0.169$ (solid blue lines represent $x_1(t)$, green dashed lines depict $x_2(t)$, black dotted lines are the safety limits).

decreasing it otherwise). After 12 seconds and 7 bisection steps we have found out that for $a^\star = 0.169$ the MPC controller avoids the unsafe sets $\mathcal{Z}_i$ for any $x(0) \in \mathcal{I}$ with $\mathcal{I} = \{x(0) \mid -a^\star \leq x_{3,4}(0) \leq a^\star,\ x_{1,2}(0) = 0\}$. Several examples of safe closed-loop trajectories are depicted in Figure 6.13. As expected from the theoretical results, all such safe trajectories avoid the unsafe sets.

Plant-Model Mismatch

Furthermore, we wish to verify whether the MPC controller, which uses (6.72) as the prediction model with $k_1 = 8.7932 \cdot 10^{-4}$ and $k_2 = 7.3772 \cdot 10^{-4}$, provides safety guarantees even when it's control actions are applied to a system with $k_2 = \widetilde{k}_2 = k_1$ (e.g. the second valve is damaged and exchanged for a valve that is identical with the first one). As discussed in Remark 6.2.17, in such a case, the open-loop sequence is no longer equal to the closed-loop one, thus the approach of Section 6.2.2 needs to be used. Specifically, we have solved (6.60) where in (6.60d) we have used the dynamics obtained for $\widetilde{k}_2$, while the KKT conditions in (6.60e) were formulated using the original prediction model. The investigated set of initial conditions was $\mathcal{I} = \{x(0) \mid -0.16, \leq x_{3,4}(0) \leq 0.16,\ x_{1,2}(0) = 0\}$, the sets of unsafe states was the

same as in the previous section and the number of examined steps $P = 20$. After 16 seconds we obtained a negative certificate. This means that the MPC controller will still preserve all of the performance requirements *ad infinitum*, even when a new valve will be embedded into the system. On the other hand, if $\widetilde{k}_1 = k_2$ is considered, then the MPC violates the performance boundaries at least from one initial condition $x(0) = [0,\ 0,\ 0.153,\ -0.04]^T$. Both results are investigated in Figure 6.14 for several randomly selected initial conditions, as well as for the one obtained from the verification.



(a) Verification with $\widetilde{k}_2 = k_1$.

(b) Verification with $\widetilde{k}_1 = k_2$.

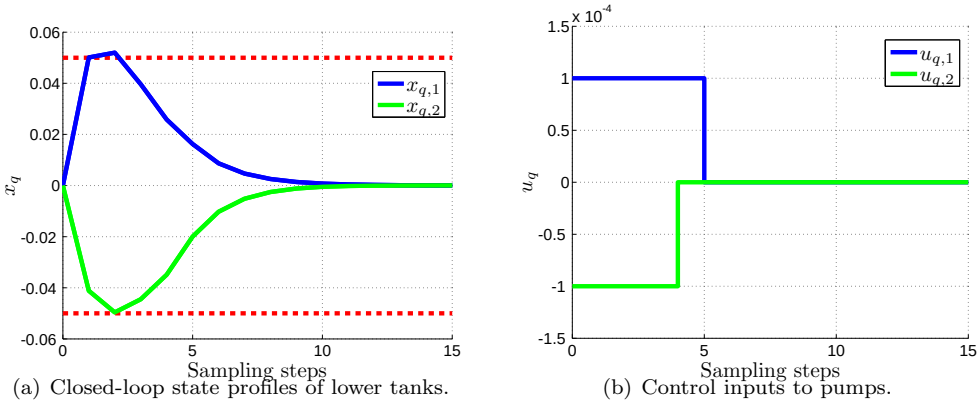
Figure 6.14: Safety verification of MPC policy with plant-model mismatch. Simulations are performed for several randomly selected initial conditions, identical for both scenarios. Closed-loop feedback from the initial condition $x(0) = [0,\ 0,\ 0.153,\ -0.04]^T$, obtained by the verification, is emphasized by the black color.

Verification of Quantized Feedback

Finally, a case study for the last scenario 3 is elaborated next. Here, our objective is to verify if MPC feedback law in (6.43) forces closed-loop state profiles to avoid each set of unsafe states $\mathcal{Z}_i$, $i = 1, \ldots, 4$ from $\mathcal{I} = \{x(t) \,|\, -0.2 \leq x_{3,4}(t) \leq 0.2,\ x_{1,2}(t) = 0\}$, while assuming that both pumps of the four tank system can change the flow rate only by 50%, i.e. their operating levels are $u_q \in \{-1, 0, 1\}$ (what corresponds to 0, 10^{-4} or $3 \cdot 10^{-4}$ $\mathrm{m^3 s^{-1}}$).

To proceed, we have constructed MPC policy (6.43) with the original setup, yet with the prediction horizon $N = 6$ to decrease the computational burden. Next,

quantization function $f_q(\cdot)$ as in (6.61) was devised. Specifically, we have computed quantization levels as all possible combinations of available pump positions $q = [(-1,-1)\ , (-1,0), (-1,1), (0,-1), (0,0), (0,1), (1,-1), (1,0), (1,1)]$ and subsequently for each combination q_i we synthesized domain $\mathcal{P}_i$ over which q_i is active as in (6.65). Then, we have imposed this quantization law for entire open-loop profile (6.69), formulated MILP in (6.70) in YALMIP for 5 verification steps and solved it iteratively (for each $\mathcal{Z}_i$, $i = 1, \ldots, 4$) by using GUROBI. After the first iteration, which took 49 seconds[8] to compute, we have received a positive answer for initial condition $x(0) = [0,\ 0,\ 0.176,\ -0.170]^T$, what indicates that the MPC policy is not safe as it forces the level in the first tank to violate the prescribed constraint $\mathcal{Z}_1$. The example is graphically illustrated in Fig. 6.15.



(a) Closed-loop state profiles of lower tanks.

(b) Control inputs to pumps.

Figure 6.15: Safety verification of MPC policy with quantized feedback law is presented. Violation of the prescribed safety constrains is shown in Figure 6.15(a), from the initial condition $x(0) = [0,\ 0,\ 0.176,\ -0.170]^T$. The associated quantized control inputs are depicted in Figure 6.15(b).

Remark 6.2.22 *The same procedure was then applied also with the largest safe bound of initialization states (for the real-valued control inputs)* $\mathcal{I} = \{x(t)\,|\, -0.169 \leq x_{3,4}(t) \leq 0.169,\ x_{1,2}(t) = 0\}$ *for which a negative certificate was obtained. Thus, we have that the aforementioned initial set* $\mathcal{I}$ *is still the largest safe bound even when quantized control law comes into play.*

[8] On a 1.7 GHz CPU running Matlab R2013a.

6.3 Application to Memory Reduction Techniques in Explicit MPC

In this section we show how the proposed verification method can be used to reduce the memory consumption of explicit MPC strategies. The main idea of this memory reduction technique is to exploit the fact that if the controlled system is assumed to be initialized only from a particular initial set $\mathcal{I}$, then storing of the entire polytopic partition is not needed since the explicit MPC policy might never employ specific regions. In another words, if region $\mathcal{R}_i$ will never be reached from the set of all initial conditions $\mathcal{I}$, then we have that it is redundant and it can be removed from the polytopic partition Ω.

6.3.1 Problem Statement

Let $\mathcal{I} \subseteq \mathbb{R}^n$ denote a set of all investigated initial conditions. Assume $\mu : \mathbb{R}^n \to \mathbb{R}^m$ to be a multiparametric solution of (6.43) in a form of

$$\mu_i(x) = F_i x + g_i \text{ if } x \in \mathcal{R}_i,\ i \in \mathcal{S}, \tag{6.74}$$

with local expressions $F_i \in \mathbb{R}^{m \times n}$ and $g_i \in \mathbb{R}^m$, set of indexes $\mathcal{S} = \{1, \dots, M\}$ and a polytopic partition $\Omega = \cup_i \mathcal{R}_i, i \in \mathcal{S}$. We aim at reducing the memory consumption of (6.74) by means of replacing it by a new optimizer $\widetilde{\mu} : \mathbb{R}^n \to \mathbb{R}^m$ defined as

$$\widetilde{\mu}_i(x) = F_i x + g_i \text{ if } x \in \mathcal{R}_i,\ i \in \widetilde{\mathcal{S}}, \tag{6.75}$$

with a set of indices $\widetilde{\mathcal{S}} \in \mathbb{R}^{\widetilde{M}}$, such that following conditions are met:

R1: $\widetilde{M} \leq M$,

R2: $\forall x(0) \in \mathcal{I}$, $x_{\mathrm{cl}}(t) \notin \mathcal{R}_i$ for all $i \in \mathcal{S} \backslash \widetilde{\mathcal{S}}$ and for all $t > 0$.

6.3.2 Memory Reduction Algorithm

Note that both optimizers (6.74) and (6.75) have their local expressions F_i and g_i, as well as regions $\mathcal{R}_i$, identical. The only difference is in the set of indexes denoting their domains, i.e. $\widetilde{\mathcal{S}} \subseteq \mathcal{S}$. Therefore, to devise a new controller $\widetilde{\mu}(x)$ we need to determine the set $\widetilde{\mathcal{S}}$ that indicates only regions that are essential during the control.

To proceed, we define a set of all unsafe states as $\mathcal{Z} = \mathcal{R}_i$ with some $i \in \mathcal{S}$. Now, to satisfy the problem R2, we need to determine whether the MPC policy (6.43), initialized from $\mathcal{I}$, forces the closed-loop trajectory $x_{\rm cl}(t)$ to enter the set $\mathcal{Z}$ for any further time instants $t > 0$. We know that the optimization problems (6.58) or (6.60), respectively, provide us the certificate for problem R2 (cf. Theorem 6.2.11 and Theorem 6.2.15), where the result may be twofold. If a negative certificate is obtained, then we have that $x_{\rm cl}(t) \notin \mathcal{Z}$ for any $t > 0$. This indicates that the region $\mathcal{R}_i$ is redundant as the control law $\mu_i(x) = F_i x + g_i$ if $x \in \mathcal{R}_i$ will never be employed. Thus, the index $i \in \mathcal{S}$ will not be incorporated into the set of all essential regions $i \notin \widetilde{\mathcal{S}}$. In another words, the regions $\mathcal{R}_i$ with control law $\mu_i(x)$ are not included in $\widetilde{\mu}(x)$. On the other hand, if a positive certificate is obtained, then $\mathcal{R}_i$ is essential as $\mu_i(x)$ might be used during the feedback. Hence, $i \in \widetilde{\mathcal{S}}$. The procedure then updates the unsafe set $\mathcal{Z}$ by e.g. a next region $\mathcal{R}_{i+1}$ and performs another certification. The algorithm terminates when all regions $i \in \mathcal{S}$ are examined and a set of all essential regions $\widetilde{\mathcal{S}}$ is yielded. Finally, the $\widetilde{\mu}(x)$ is constructed via set of indexes $\widetilde{\mathcal{S}}$. The entire procedure, which provides answer for problem R1 and R2, is described in Algorithm 10 and its efficiency will be reported in a sequel.

Algorithm 10: Memory reduction based on reachability verification.

Input: $\mu(x)$ as in (6.74), $\mathcal{I}$

Output: $\widetilde{\mu}(x)$ as in (6.75)

Initialization: $\widetilde{\mathcal{S}} \leftarrow \emptyset$;
for $i = 1, \ldots, M$ **do**
  $\mathcal{Z} \leftarrow \mathcal{R}_i$;
  Solve (6.58)/(6.60);
  if *Certification is positive* **then**
    $\widetilde{\mathcal{S}} \leftarrow \widetilde{\mathcal{S}} \cup i$;
  end
end
return $\widetilde{\mu}_i(x) = F_i x + g_i$ *if* $x \in \mathcal{R}_i,\ \forall i \in \widetilde{\mathcal{S}}$;

Remark 6.3.1 *Obviously, since the objective is to reduce the memory of $\mu(x)$, we would like to problem R1 to be be given be a strict inequality, i.e. $\widetilde{M} < M$. Yet, the proposed reduction technique can not enforce such criteria as all of the regions $\mathcal{R}_i$, $i \in \mathcal{S}$ may be reachable from $\mathcal{I}$, thus $\widetilde{M} = M$.*

6.3.3 Illustrative Example (Double Integrator)

To provide an illustrative example that demonstrates the efficiency of the proposed memory reduction technique, we will consider a car which dynamics are described by a following LTI system in discrete-time domain

$$x(t+1) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} 1 \\ 0.5 \end{bmatrix} u(t), \tag{6.76}$$

where $x_1(t)$, $x_2(t)$ and $u(t)$ denote the cars position, speed and acceleration, respectively. Consider (6.76) be subjected to

$$-10 \leq x(t) \leq 10, \tag{6.77a}$$

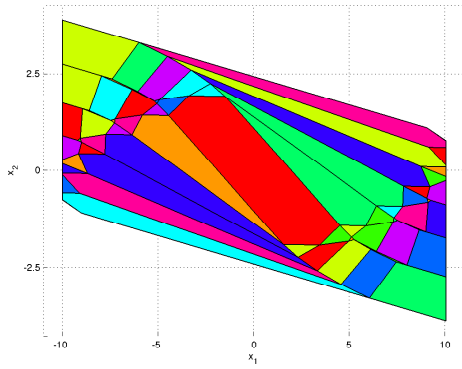
$$-1 \leq u(t) \leq 1. \tag{6.77b}$$

The control objective is to drive the car to the origin while satisfying all of the aforementioned constraints. To proceed we have constructed MPC strategy as in (6.43) with model (6.76), constraints (6.77), $N = 5$, $Q_x = \mathrm{diag}(1, 1)$, $Q_u = 1$, Q_N being the solution of the algebraic Riccati equation, $\mathcal{X}_f$ being the constraint admissible set of the plant in closed-loop with the LQR controller, obtained for the same cost function of the MPC. Subsequently, we have solved this problem via mp-QP what after 3 seconds yielded an explicit MPC as in (6.74) defined over polytopic partition $\Omega = \cup_i \mathcal{R}_i$, $i = 1, \ldots, M$ which consisted of $M = 65$ regions. The polytopic partition is depicted in Figure 6.16. We have assumed that the system can be initialized only from the set $\mathcal{I} = \{x(0) \mid -10 \leq x_1(0) \leq 10, x_2(0) = 0\}$ what corresponds to the scenario when the car is stationary and may attain an arbitrary position (e.g. car is parked anywhere in the parking spot).

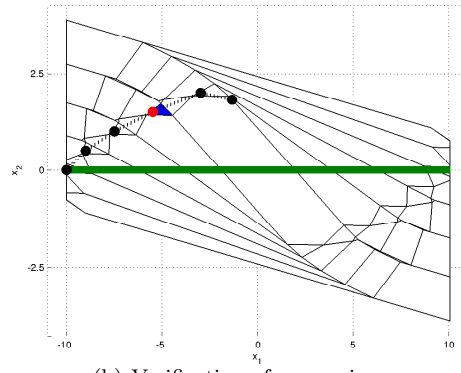
In order to reduce the memory of (6.74), we have applied Algorithm 10. Since the MPC policy have been devised to satisfy $X_{\mathrm{ol}} = X_{\mathrm{cl}}^N$ per Remark 6.2.7, thus the algorithm has employed verification procedure in (6.58). After 59 seconds all essential regions were determined and a new explicit MPC optimizer $\widetilde{\mu}(x)$, defined over 25 regions, was constructed. In another words, we have that only 25 regions are essential and other 40 regions are redundant as they will never be used during the control. Thus, by removing these regions we have reduced the complexity of $\mu(x)$ by a factor of $\triangle = 65/25 = 2.6$.

This procedure is illustrated in Figure 6.16. Here, Figure 6.16(a) shows the complex polytopic partition of $\mu(x)$, composed of 65 regions. Figure 6.16(b) depicts the case where the verification technique (6.58) determined that the region

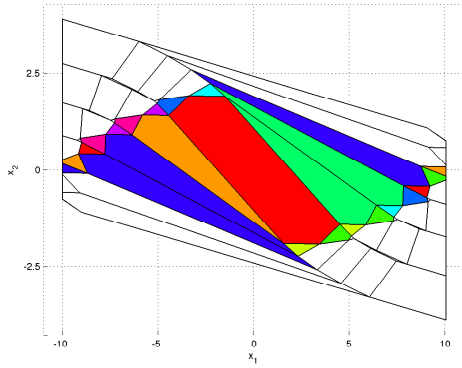
$\mathcal{R}_9$ (blue polytope) is reachable from $\mathcal{I}$ (green polytope) within 3 steps. Next, Figure 6.16(d) illustrates the verification of 20 equidistant initial conditions from $\mathcal{I}$ and Figure 6.16(c) shows the polytopic partition of the reduced optimizer $\widetilde{\mu}(x)$.



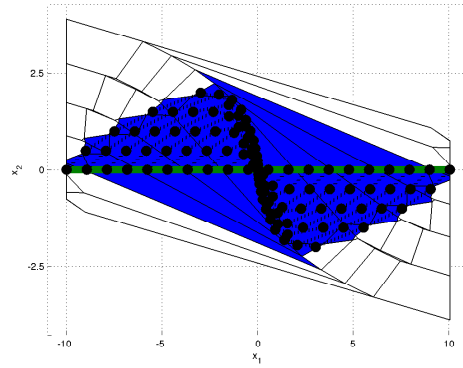
(a) Complex polytopic partition of $\mu(x)$.



(b) Verification of one region.



(c) Reduced polytopic partition of $\widetilde{\mu}(x)$.



(d) Closed-loop dynamics.

Figure 6.16: Illustration of memory reduction technique.

Moreover, we have applied the proposed memory reduction technique for various predictions horizons $N = 1, \ldots, 10$. Results are compactly reported in Table 6.4. The reduction factor is defined by $\triangle = \frac{|\widetilde{\mathcal{S}}|}{|\mathcal{S}|}$, where $|\cdot|$ denotes the cardinality. Here, an interesting observation can be made. Even thought that the prediction horizon increases $N > 5$, the number of regions $|\widetilde{\mathcal{S}}|$ remains the same. This is due to the fact that further generated regions are not reachable from $\mathcal{I}$.

Remark 6.3.2 *It should be noted that the main advantage of the proposed verifi-*

N	$\|\mathcal{S}\|$	$\|\widetilde{\mathcal{S}}\|$	$\triangle$	Time [sec]
1	9	3	3.00	6.89
2	19	5	3.80	12.83
3	33	17	1.94	23.32
4	45	23	1.96	36.48
5	65	25	2.60	59.42
8	97	25	3.88	125.73
10	109	25	4.36	176.79

Table 6.4: Results of the memory reduction technique.

cation method is that the reachability/unreachability status of a particular critical region can be determined without having to know the full explicit solution. This opens up the possibility to combine the proposed method with a parametric programming solver to directly generate a simple solution, without the need to construct the full (complex) explicit optimizer first.

6.4 Conclusions

In this Chapter, we have introduced two methods that verify properties of MPC strategies. The first approach exploited parametric programming to construct explicit solution of MPC and to derive performance boundaries, each of which represented certain control property. Namely, recursive constraint satisfaction, closed-loop stability and deterioration of performance. Associated quantized version of the PWA feedback law was then found by applying Voronoi diagram technique. Eventually, certification algorithm was proposed, which was used determined if the quantized controller violates performance bounds. The second approach has taken different road. Here, KKT conditions were employed to characterize optimal control inputs, hence explicit solution was no longer needed. The certification problem was formulated as MILP, where reachability of unsafe states have been verified. Application of both methods was demonstrated on several examples. Results are illustrated and documented in tables.

Part III

Practical Contributions

Chapter 7

pH Control in Chemical Vessel

Maintaining a specific value of pH in chemical and technological processes is vital in order to satisfy quality requirements of final products. This is important in water treatment facilities where value of pH greatly effects the quality of water purification (Qian et al., 2014). It has been reported, that bad pH conditions result in production of bacteria dangerous to human health. (Qin et al., 2006) describe how pH condition affects the coagulation process in the treatment of reservoir water which has then impact on purity of the water. Biochemical experiments are another application where regulation of a specific value of pH is needed. Here, unfavorable pH conditions negatively effect entire experiments (Misiewicz et al., 2015). pH plays a vital role in medical research and medicine preparation since vast majority of all drug preparation requires specific pH conditions (Georgiev et al., 2013). Several scientific papers are dealing with a pH control or with a neutralization control. Since the behavior of the neutralization process is highly non-linear, controller design via conventional means has been proven to be insufficient (Ibrahim, 2008) in several cases. Despite the fact, a simple PID controller is often used in order to control the pH value.

In this thesis, we aim to improve the behavior of the PID controller by introducing another layer of control based on an optimization. Such an approach is often used in chemical and petrochemical industry with the goal of increasing the overall production. The optimization layer consist of a model prediction controller, which serves as a supervisor controller to the PID controller. In such a setup the

MPC shapes the setpoint for PID controllers in optimal fashion. The advantage of introducing the MPC to the control scheme is that we will achieve optimal performance of the plant, i.e., technological constraints are satisfied while a specific performance criterion is minimized. In literature, this relation of MPC and PID controller are called a reference governor control (Bemporad, 1998; Borrelli et al., 2009).

7.1 Experimental Process

7.1.1 Process Equipment

The process consists of a continuously stirred tank which is fed with two streams of solutes. The first stream is an acid solution with flow rate F_{A} and concentration of c_{A} and the second stream is a base solution with flow rate F_{B} and concentration of c_{B}. Transfer of these solutions, from the storage tanks to the stirred tank, is ensured by two identical pumps P_{A} for acid and P_{B} for base. Finally, the output signal is detected via pH probe. The scheme of the process is depicted in Figure 7.1 and parameters of each component are given by:

- Stirred tank with volume of $1.5\,\mathrm{dm}^3$.
- Two peristaltic pumps (P_{A}, P_{B}) each of which admits voltage of (U_{A},U_{B}) in range of $[0, 5]\,\mathrm{V}$, what corresponds to the flow rate (F_{A}, F_{B}) between $[0, 12]\,\mathrm{ml\,s}^{-1}$.
- Two tanks each of which can store up to $100\,\mathrm{dm}^3$ of solution.
- pH probe which returns signal in range of $[0, 5]\,\mathrm{V}$.

7.1.2 Chemical Experimental Setup

Chemicals which will be considered in this work are the acetic acid (CH_3COOH) and the sodium hydroxide (NaOH). Both solutes are used to create their solutions with concentration of $c_{\mathrm{A}} = 0.01\,\mathrm{M}$ and $c_{\mathrm{B}} = 0.01\,\mathrm{M}$, respectively. Since the molar mass of NaOH is equal to $40\,\mathrm{g}$ and CH_3COOH to $60\,\mathrm{g}$, this means that in order to prepare the base solution one need to dissolve $40\,\mathrm{g}$ of NaOH in $100\,\mathrm{dm}^3$ of H_2O, and for the acid solution we need $60\,\mathrm{g}$ of CH_3COOH in $100\,\mathrm{dm}^3$ of H_2O.

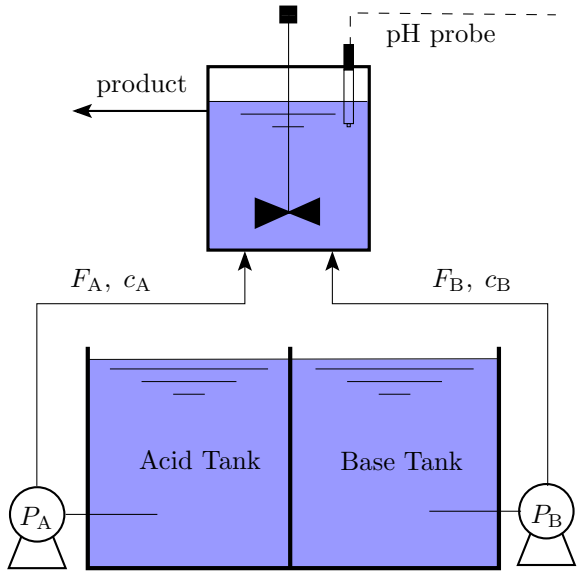


Figure 7.1: Illustration of neutralization reaction vessel

The dissociation reaction of used solutes in water is given by

$$\mathrm{NaOH} \xrightarrow{\mathrm{H_2O}} \mathrm{Na^+} + \mathrm{OH^-}, \tag{7.1a}$$

$$\mathrm{CH_3COOH} \xrightleftharpoons{\mathrm{H_2O}} \mathrm{H^+} + \mathrm{CH_3COO^-}. \tag{7.1b}$$

Here, one can notice that since the sodium hydroxide is strong base it dissociates in water completly (7.1a), contrary to the acetic acid, which is a weak acid, dissociates only partially (7.1b). This means that in the water solution there exists both HAc molecules as well as $\mathrm{CH_3COO^-}$ molecules. The pH of the base solution is thus $\mathrm{pH} = 12$ and by introducing dissociation constant of the acetic acid $k_\mathrm{a} \approx 10^{-5}$ the pH value of the acid solution is $\mathrm{pH} = 3.5$.

By feeding both solutions into the reaction vessel the following chemical reaction takes place

$$\mathrm{NaOH} + \mathrm{CH_3COOH} \xrightleftharpoons{\mathrm{H_2O}} \mathrm{NaCH_3COO} + \mathrm{H_2O}. \tag{7.2}$$

Since the final product has no electrical charge, as well as reactants, electroneutrality equation has to be considered, which can be written as follows[1]

$$[\mathrm{Na^+}] + [\mathrm{H^+}] = [\mathrm{OH^-}] + [\mathrm{CH_3COO^-}]. \tag{7.3}$$

[1]We note that in all chemical equations, $[\cdot]$ denotes concentration, e.g. $[\mathrm{H^+}]$ stands for the concentration of hydrogen ions, which is expressed in $\mathrm{mol/m^3}$ or simply M.

The value of pH in the mixing tank can be then detected by the concentration of hydroxide cations as

$$\mathrm{pH} = -\log_{10}\left([\mathrm{H}^+]\right). \tag{7.4}$$

Remark 7.1.1 *To compute pH analytically is not an easy task. This problem boils down to following cubic algebraic equation*

$$[\mathrm{H}^+]^3 + (x_B + k_a)[\mathrm{H}^+]^2 + (x_B k_a - x_A k_a)[\mathrm{H}^+] - k_w k_a = 0,$$

where $k_w \approx 10^{-14}$ *and* $k_a \approx 10^{-5}$ *are dissociation constants of the water and the acetic acid respectively,* $x_A = [\mathrm{CH_3COOH}] + [\mathrm{CH_3COO^-}]$ *is the acetic acid concentration (in the mixing tank) and* $x_B = [\mathrm{Na}^+]$ *is concentration of the sodium cation (in the mixing tank). Moreover, by incorporating to the previous cubic equation also material balance of the system*

$$V\frac{dx_A}{dt} = F_A c_A - (F_A + F_B)x_A,$$
$$V\frac{dx_B}{dt} = F_B c_B - (F_A + F_B)x_B,$$

one obtains a mathematical model of the process. Yet, for the purposes of this work we will consider much simpler experimental model that will be derived in Section 7.2.

7.1.3 Control Setup

Assume that both (the acid solution and the base solution) streams are pumped into the tank, where are continuously mixed by the stirrer. The final product is taken from the top of the vessel, such that the volume of the mixture remains the same. The measuring probe is placed directly next to the product outlet and the flow rate of the acid solution is maintained constant, i.e. $\dot{F}_{\mathrm{A}} = \dot{U}_{\mathrm{A}} = 0$. The control objective is to manipulate the voltage U_{B} of the pump P_{B} such that pH value will track the desired reference. Considered variables are summarized next:

1. The controlled output is pH value in the stirring tank, $\bar{y} \in \mathbb{R}$, where $\bar{y} \in [0, 14]\mathrm{pH}$.

2. The control input is voltage in the pump P_{B}, $\bar{u} := U_{\mathrm{B}} \in \mathbb{R}$, with $\bar{u} \in [0, 5]\,\mathrm{V}$.

3. Constants are:

- Voltage in the pump P_A, $U_A = 2.5\,\mathrm{V}$ ($F_A \approx 6\,\mathrm{ml\,s^{-1}}$).
- Concentration of the acetic acid solution: $c_A = 0.01\,\mathrm{M}$.
- Concentration of the sodium hydroxide solution: $c_B = 0.01\,\mathrm{M}$.

7.2 Model of the Process

In Section 3.2.2 we have stated that a good mathematical model is a cornerstone of MPC. This is due to the fact that if the mathematical model does not resemble the controlled plant, then the predictions of MPC would be incorrect, what eventually results in a poor control performance (see e.g. Figure 3.2). However, we need to keep in mind, that the accuracy comes hand in hand with the complexity of the prediction model. It is therefore important to derive such model, which on the one hand tracks the real-process, but on the other hand keeps its description simple. Or in other words, we need to find the best trade-off between the complexity and the precision of the model.

Calibration Function for the pH probe

In order to have a better indication of the pH signal from the probe, one need to create a calibration function $f_{\mathrm{pH}}(\cdot) : \mathbb{R} \to \mathbb{R}$, which maps voltage into pH value. This can be done e.g. by comparing the voltage signal of the pH probe with the prescribed pH values of buffers. In our experiment, we have used buffers with pH equal to $\{4, 7, 10\}$ for which the probe showed corresponding signals $U_{\mathrm{pH}} = \{1.340, 2.615, 3.825\}\,\mathrm{V}$. By approximating these points with polynomial of the first degree, based on the least square criteria, we have obtained the probe's calibration function defined as

$$f_{\mathrm{pH}}(U_{\mathrm{pH}}) = 2.41 U_{\mathrm{pH}} + 0.739,$$

where U_{pH} is signal from the pH probe (in volts). From now on we define $\bar{y} := f_{\mathrm{pH}}(U_{\mathrm{pH}})$. The same procedure could be also applied to each peristaltic pump (i.e. to determine the relationship between F_A and U_A, or F_B and U_B). Yet, in this work we decided to operate with the voltage signal instead.

Step Responses

The identification task has been approach by a black box technique (see Remark 3.2.2), hence we have assumed that we have knowledge of the process dynam-

ics and that this information can be determined via relations between the input and output signals, i.e. via step responses. To achieve this goal, we have firstly steered the system to the steady state by using setup described in Section 7.1.3 and by applying control input $\bar{u}^s = 2.5\,\mathrm{V}$. The associated steady-state output was $\bar{y}^s \approx 7$. Subsequently, we have performed 6 step responses on the input signal $\bar{u}_i = \{3.0, 3.5, 2.5, 2.0, 1.5, 2.5\}$ volts, where the frequency of steps was set to 1500 seconds. The corresponding state and input profiles are depicted in Figure 7.2 and are denoted by blue and green color respectively.

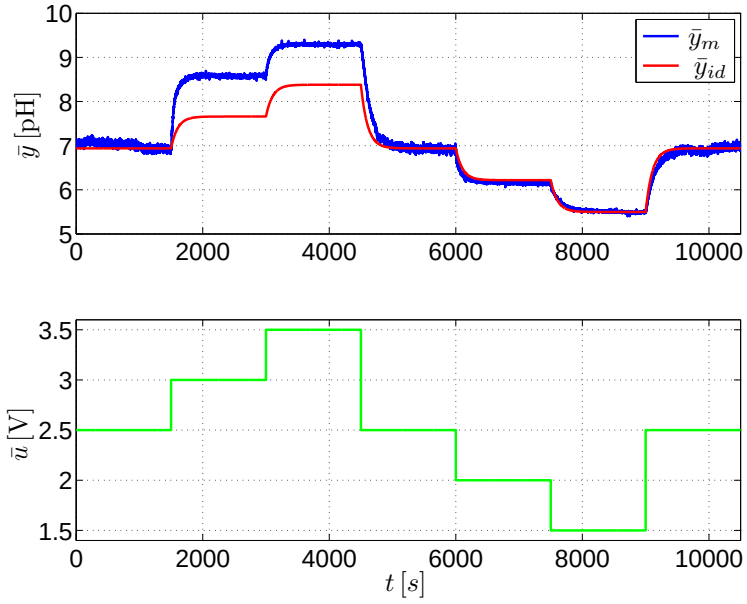


Figure 7.2: Identification data of the process.

Model Identification

With the measured data in hand, our next objective is to derive a simple mathematical model which will describe systems dynamics with an acceptable accuracy. To proceed, we have performed normalization of the measured data, i.e. we have separated all step responses, divide each of them by the corresponding step change $\triangle u_i = \bar{u}_i - \bar{u}^s$, summarize these steps and dividing them with their quantity. This led to the normalized step response, which gave us a better insight into the process

dynamics. Subsequently, based on this normalized step response, we have identified the process as the system of the fist order. The corresponding continuous-time dynamics is

$$105\dot{y}(t) + y(t) = 2.803u(t), \tag{7.6}$$

where y and u are output and input values in the deviation form, i.e. $\bar{y} = y + \bar{y}^s$ and $\bar{u} = u + \bar{u}^s$.

To determine the accuracy of the model (7.6), we have taken the same sequence of the control inputs as in Figure 7.2 and passed it by the identified dynamics (7.6). The corresponding output profile is denoted by the red color in Figure 7.2. We can observe that the controlled process is nonlinear as it exhibits non-symmetrical step responses. Due to this nonlinear nature of the system, our identified (linear) model does not accurately capture dynamics, when the product attains alkaline values, i.e. above pH = 7. However, this differences can be compensated by the designed controller.

Finally, the model in (7.6) has been discretized, what led into differential equations in form of

$$y(t) = m_1 u(t) + m_2 u(t-1) + m_3 y(t-1), \tag{7.7}$$

where $u(t-1)$ represents the control input from the previous sampling instant.

7.3 PID Controller

Nowadays, PID controllers are the most commonly used control algorithms in the industry, where this control strategy utilizes more than 90% of all applied feedback loops (Desborough and Miller, 2002). This is mainly due to their structural simplicity, robustness, low computational demands, easy design and straightforward tuning. An exhausting PID theory can be find in (ström and Hägglund, 1995).

It is well know that there are several PID implementation schemes (e.g. parallel, standard, or serial formulation) and multiple techniques how to translate PID controller in continuous domain into discrete one (see for example (Bobál et al., 2006)). In our work, we consider PID controller in the standard decomposition, depicted in Figure 7.3, in form

$$\begin{aligned} u(t) =& u(t-1) + K\left(1 + \frac{T_s}{2T_i} + \frac{T_d}{T_s}\right) e(t) + \\ & K\left(-1 + \frac{T_s}{2T_i} - 2\frac{T_d}{T_s}\right) e(t-1) + K\left(\frac{T_d}{T_s}\right) e(t-2), \end{aligned} \tag{7.8}$$

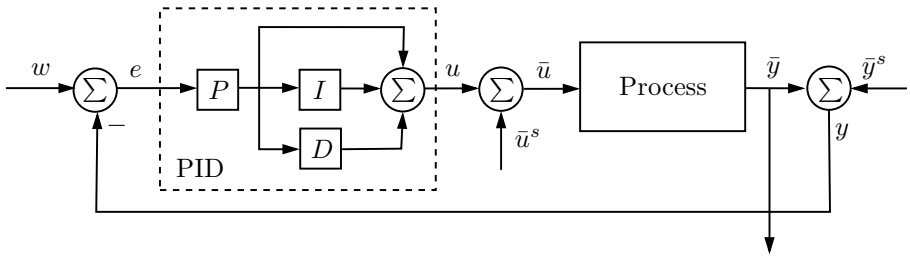


Figure 7.3: Closed-loop with PID controller.

with parameters K, T_i and T_d denoting PID gain, integral time constant and derivative time constant, respectively, which are associated with P, I and D part of PID structure. Moreover, T_s is the sampling time and e.g. $e(t-1)$ denotes value of the control error in the previous time instant $t - T_s$.

To design a PID controller we need to determine parameters K, T_i and T_d in (7.8). There are numerous techniques which suggest how one can synthesize and tune a PID controller (O'Dwyer, 2009), majority of which is already embedded in various toolboxes (Oravec and Bakošová, 2012; The MathWorks, 2016b). In this work we have used an interactive toolbox for PID design called pidtuner (The MathWorks, 2016a). The parameters K, T_i and T_d have been chosen such that the closed-loop feedback, with identified model (7.7), was fast and robust. The final values of all parameters of the PID controller were chosen as follows

$$K = 1.5, \quad T_i = 21, \quad T_d = 0. \tag{7.9}$$

By plugging parameters (7.9) into (7.8) the differential equation of the design controller is

$$u(t) = u(t-1) + 1.536e(t) - 1.464e(t-1). \tag{7.10}$$

Moreover, since the D-part of the PID controller is turned off, by $T_d = 0$, we have that (7.10) is a PI controller. For brevity, let us abbreviate (7.10) as $u(t) = p_1 u(t-1) + p_2 e(t) + p_3 e(t-1)$.

7.4 MPC Reference Governor

With the PI controller in had, we will show that one can enhance its performance even further. It is known that PID controllers generate control inputs only based on

their formula e.g. as in (7.8),i.e. without taking (a-priori) into account any type of constraints. This problem is usually handled via imposing additional blocks into control schemes e.g. such as subsequent saturation of the generated control inputs. However, terms like minimization of power consumption or specific safety conditions are not an easy task to include (if even possible) into the control design.

On the other hand, MPC excels in providing the aforementioned requirements. The concept of MPC strategy was introduced in Section 3. In this work, the MPC policy will be exploited as a reference governor to the PI controller, which was designed in the previous Section 7.3, i.e. as to shape the reference w such that the inner PID-loop will satisfy all of the prescribed constraints and the outflow pH value from the process $\bar{y}$ will track the targeted reference r. Such reference governor control scheme is depicted in Figure 7.4.

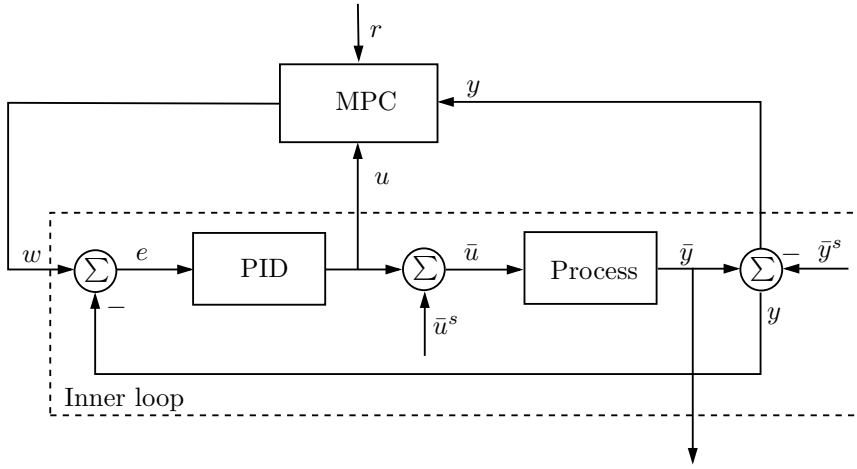


Figure 7.4: Scheme of the reference governor.

7.4.1 MPC Formulation

In this work, we consider MPC to be given as a following optimization problem

$$\min_{w_0,\dots,w_{N-1}} Q_{\mathrm{s}}s^2 + \sum_{k=0}^{N-1} \left(Q_{\mathrm{r}} \left(w_k - y_k\right)^2 + Q_{\mathrm{w}}\Delta w_k^2 + Q_{\mathrm{y}} \left(y_k - r\right)^2 \right) \tag{7.11a}$$

$$\text{s.t. } u_k = p_1 u_{k-1} + p_2(w_k - y_k) + p_3(w_{k-1} - y_{k-1}), \tag{7.11b}$$

$$y_k = m_1 u_k + m_2 u_{k-1} + m_3 y_{k-1}, \tag{7.11c}$$

$$\Delta w_k = w_k - w_{k-1}, \tag{7.11d}$$

$$u_{\min} \leq u_k \leq u_{\max}, \tag{7.11e}$$

$$w_{\min} \leq w_k \leq w_{\max}, \tag{7.11f}$$

$$y_{\min} - s \leq y_k \leq y_{\max} + s, \tag{7.11g}$$

$$s \geq 0, \tag{7.11h}$$

with constraints (7.11b)-(7.11g) enforced for all $k = 0, \dots, N-1$, where N is the prediction horizon. Denote w_k the shaped reference, $y_k \in \mathbb{R}$ the output from the process and $u_k \in \mathbb{R}$ the control input from the PID controller at the k-th predicted step. Moreover, $r \in \mathbb{R}$ is the desired reference, $s \in \mathbb{R}$ is a slack variable and Q_{s}, Q_{y}, Q_{w}, Q_{r} are positive definite weighting coefficients. Optimization problem in (7.11) is initialized by the vector of parameters $\theta(t) := [y(t-1), y(t-2), w(t-1), w(t-2), u(t-1), r(t)]^T \in \mathbb{R}^6$, where e.g. $y(t-1)$ and $y(t-2)$ is the output from the process delayed by one and two sampling instants T_s, respectively. The vector of optimized variables is given by $z := [w_0, \dots, w_{N-1}, s]^T \in \mathbb{R}^{N+1}$.

Each part of the objective function and every constraint in (7.11a) has following impact on the optimization problem. The first term penalizes a violation of the output constraints in (7.11g), the second term provides the convergence of the process output to the shaped reference, the third term penalizes the fluctuation of the shaped reference, which is in the last term forced to track the real reference. The first constraint (7.11b) represents the dynamics of the PI controller (7.10) and (7.11c) simulates evolution of the controlled process as in (7.7). Next (7.11d) defines fluctuation of shaped reference, positiveness of slacks is given by (7.11h) and finally (7.11e)-(7.11g) restrict values of predicted inputs, shaped references and outputs, respectively.

Parameters of MPC

Here, we will define all parameters of the MPC optimization problem in (7.11). Let us start from the objective function. The weighting matrices in (7.11a) have been firstly chosen based on the simulation control performance, while the a-posteriori tuning has taken place during real experiments. The final values of weighting matrices are

$$Q_{\mathrm{s}} = 10^4, Q_{\mathrm{r}} = 1, Q_{\mathrm{w}} = 10, Q_{\mathrm{y}} = 20.$$

Next, dynamics of the PID controller (7.11b) and the model (7.11c) are given by (7.10) and (7.7), respectively. In order to allow MPC policy to operate only with the admissible voltage range $\bar{u} \in [0, 5]\,\mathrm{V}$, we have defined constraints in (7.11e) with $u_{\min} = -2.5$ and $u_{\max} = 2.5$. The shaped reference w in (7.11f) was allowed to attain all possible values of the pH, i.e. $w_{\min} = -7$ and $w_{\max} = 7$, what corresponds to pH $\in [0, 14]$. Similarly, the output values in (7.11g) have been subjected to $y_{\min} = -7$ and $y_{\max} = 7$, yet soften by the slack variable s, since the identified model in (7.7) only approximates the real (nonlinear) process dynamics. The chosen prediction horizon, which still showed good control performance, was $N = 3$.

Explicit Representation of MPC

Since the optimization problem in (7.11) has quadratic objective function and linear constrains, it can be rewritten into a standard QP formulation as in (4.20). As shown in Section 4.1.3, the parametric solution of (4.20) leads to an explicit optimizer (4.33) in a form of a continuous polytopic PWA function with partition $\Omega = \cup_i \mathcal{R}_i$ consisting of $i = 1, \ldots, M$ polyhedral regions $\mathcal{R}_i$ as in (4.32) (cf. Theorem 4.1.8).

In our case, by solving (7.11) via parametric programming in MPT3 toolbox, we have obtained explicit MPC policy defined over $\Omega \subset \mathbb{R}^6$, formed by $M = 241$ regions. This, relatively large number of regions, is caused due to the fact that unnecessary large parameter space has been explored. To see this, we have to firstly realize that in the optimization problem (7.11) there are no restriction to any parameter of θ. However, e.g. the first parameter $\theta_1 = y(t-1)$, corresponding to the process output (pH) that is delayed by one sampling instant, can attain values only from the interval $[-7, 7]$ (deviation values). And the same principle applies even for all other parameters in θ. Therefore, we have attached additional constraints $-\alpha \leq \theta \leq \alpha$, with $\alpha = [7, 7, 7, 7, 2.5, 7]$, to (7.11). This eventually led

to reduction of the parameter space, where the polytopic partition was composed only by 47 regions. Finally, this simple explicit MPC controller was then exported to a self-contained file, where the sequential search point location problem (see Algorithm 4) was applied to allocate the active region, and used as a reference governor in the real experiment.

7.5 Experimental Results

In the sequel, we show experimental results of pH control in mixing vessel, where chemical reaction between strong base (solution of sodium hydroxide) and weak acid (solution of acetic acid) took place (7.2). We recall that their solutions (7.1) were prepared by the table values to attain concentration of 0.01 M, as discussed in Section (7.1.2). The controlled variable was the voltage to the pump P_{B}, which influenced the inflow of base solution, while the inflow of acidic acid solution was kept constant with P_{A} pump voltage of 2.5 V.

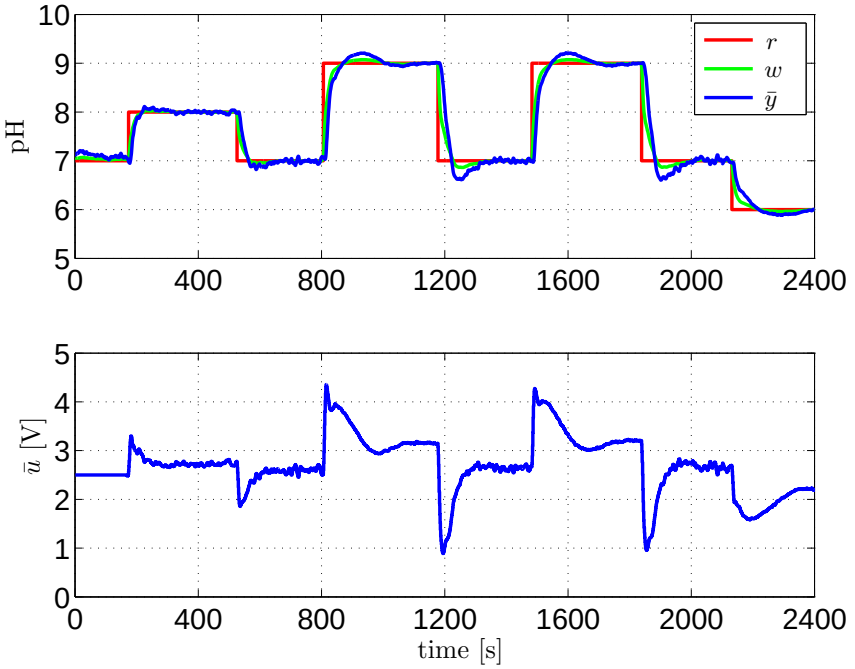


Figure 7.5: Control performance of reference governor MPC.

In Figure 7.5 is depicted performance of pH control with PI controller[2] (7.10), which was supervised by MPC-based reference governor (7.11) as it is shown in Figure 7.4.The process was initialized from the steady state $\bar{y}^s = 7$pH, given by control input $\bar{u}^s = 2.5$ V. The first reference step change was performed to pH = 8. Here, we can observe that MPC policy, based on the given step, has shaped references to PI controller (green line), in such way that controlled variable nearly did not exceeded reference, i.e. overshoot was almost eliminated. The same control performance can be seen also in the opposite direction. Next, four changes on the reference follows from pH = 7 to pH = 9 and vise versa. The increased gain of steps had a direct consequence on control performance. Particularly, to longer settling time and greater overshoots/undershoots. To see how w influenced the controlled variable, we can notice the first negative step change, which occurred at time 1080 seconds. Here, the shaped references w firstly followed the desired reference r (denoted by red color) in order to converge to the set point as fast as possible. The slope has changed, however, when controlled output reached pH = 8 and from this point forward, references w were used to mitigate undershoot and subsequent overshoot. The last step was performed to pH = 6, where the settling time was twice as large as in the reversed step direction. Generally, control below pH = 7 is much slower and requires more time to reach the reference. The reason behind this stems from the nonlinear nature of the controlled system. To see this, notice that more acidic values are associated with smaller gain of the processes, as is shown in Figure 7.2.

Figure 7.6 shows comparison between two control schemes. The first one represents standard closed-loop with PI controller (solid blue line) depicted in Figure 7.3. The second one is the reference governor scheme (solid green line), the performance of which we have already elaborated in the previous Figure 7.5. We note that both experiments were carried out in a row, with the same solutions and with step frequency 400 seconds. For illustrative purposes, shaped references w is excluded from the figure as its influence was already demonstrated. In Figure 7.6 we can observe that reference governor shaped references to PI controller, what led to less aggressive control behavior, and eventually to constraints satisfaction and smaller overshoots/undershoots, compared to standard closed-loop with PI controller. Particularly, in the first step change, PI controller had overshoot

[2]We would like to note that anti-windup technique was not applied.

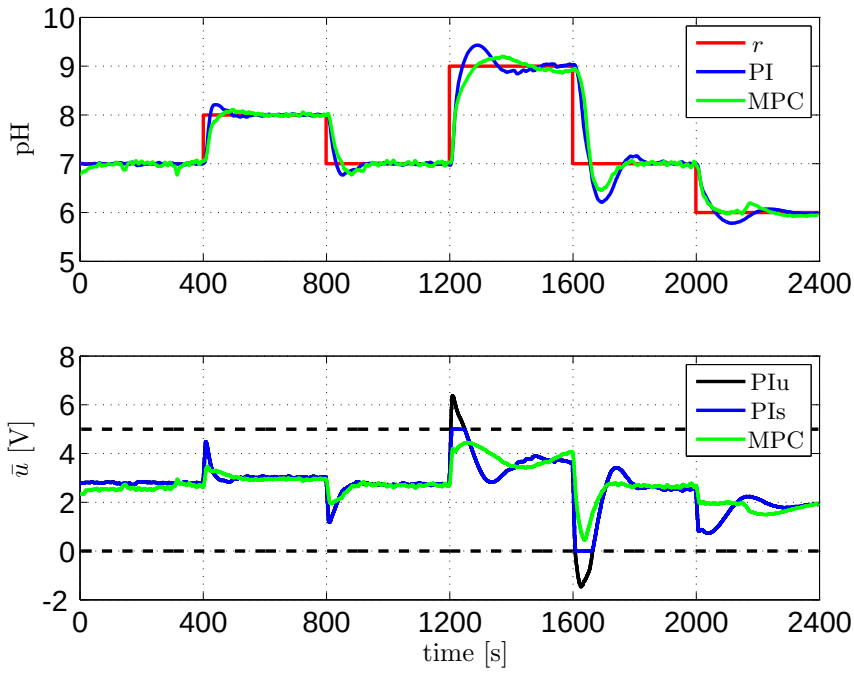


Figure 7.6: Comparison of two control schemes depicted in 7.3 and 7.4, respectively. Here, to demonstrate the control input violation, we present unsaturated control inputs (PIu), saturated control inputs (PIs) of the PI controller. Green line denotes inputs/outputs of process when MPC policy was employed. Moreover, black-dashed line denotes input constraints and red line tracking reference.

21%, while MPC policy had only 10%. Moreover, to demonstrate consumption of the control medium (solution of sodium hydroxide), we have used integral squared error (ISE). Based on this quality indicator, reference governor scheme was able to decrease medium (also energy) consumption by 1%. However, the cost of this improvement is increased settling time, which was 20 seconds[3] longer (in the first step changes) as with PI closed-loop. From the aplication point of view, we can make another interesting observation. In the last step, MPC reached desired reference much quicker then PI closed-loop. Here, disturbance occurred, which was

[3]The settling radius, around the desired reference, was considered 0.1pH.

caused due to reaction sediment inside of the base tank.

7.6 Conclusions

In this chapter, we have aimed to control value of pH in mixing vessel, where reaction between acidic acid and sodium hydroxide occurred. Our objective was to design a control strategy, which will improve performance of commonly used PI controllers. To achieve this goal, we have proposed reference governor scheme, where MPC policy provided references to PI controller. This solution was given in explicit form, thus computational burden were minimized and the requirement of optimization solver was removed. Based on the experimental results, we have shown that this way we are able to influence the control performance of PI in such manner that overshoots and undershoots are mitigated and input constraints are preserved. Therefore, by using this approach we can not only to decrease stress of the control pumps, hence to prolong their life cycle and maintain continuity of the process, but also to increase overall production.

We would like to note, that this work represents a cornerstone for our future research, where we would like to implement various advanced control strategies (e.g. disturbance modeling, anti-windup, prediction of references). From the explicit MPC point of view, we will aim to exploit proposed memory reduction techniques, presented in this thesis, which will help as to increase performance of MPC controller. For example, we will be able to increase prediction horizon in (7.11) and subsequently mitigate memory footprint of explicit solution. Subsequently, we would like to implement these simple controllers into standard industrial hardware e.g. PLC.

Chapter 8

MPC Application to Wind Turbine Generator

One of the leading problems of today society is increasing global energy consumption, what inherently forces researchers to improve the efficiency and to search for new possibilities of energy production. A remedy for this issue has been found in renewable energy sources, which in the past two decades made a great economical and political impact and revolutionized the world energy production. Its popularity arises mainly from the struggle for reduction of fossil fuels dependency, creating now economic opportunities and last, but not least, as an answer for addressing climate changes. Nowadays, the nominal production of the green energy sources is estimated to 22% of the overall world energy production (REN21, 2016) and this trend has exponentially increasing character.

The most exploited energy source (among renewables) is the wind, what makes wind turbines appealing to further research. During the last 20 years a lot of attention was dedicated to development of wind turbines, where the main goals were to maximize energy production, improve energy quality and minimize costs of installation and maintenance (e.g. via prolonging life cycle). Wind turbines use blades to convert wind flow into mechanical energy, which is then passed to the generator and eventually transformed into electrical energy. The production of energy grows with the power of wind of up to so-called rated operating point, which makes the wind turbine a cost efficient system from the perspective of components dimen-

sions and typical winds for a specific location. Modern wind turbines nowadays have their rated power up to 8 MW and are placed in various locations, e.g. with high altitudes or offshore, in order to capture stronger wind, hence to maximize production. Installations of wind turbines have ever increasing trend and today world installed wind energy capacity has reached 432 GW (GWEC, 2016).

There are two operating modes of the wind turbine, which are related with the wind power. The first mode denotes low wind region, where the aim is to capture entire wind potential by keeping the wind turbine at the point of maximum power conversion efficiency for given wind speed. In the second mode, high wind speed region, the objective is to maintain rated power output constant while reducing the aerodynamic torque to rated value and preserving nominal speed of the generator, i.e. to protect wind turbine system via sacrificing certain potential of wind power (Johnson, 2004). First operation mode is achieved by generator control, and second mode by blade pitching mechanism. This mode represents 98% of the overall operation time of the wind turbine. However, if the wind speed is dangerously too high (cut-out speed), than blades are pitched to breaking position (90 degrees) and the system is shut down. On the other hand, if the wind speed is too low (cut-in speed), then wind turbine is turned off as the insufficient to generate energy which is required to maintain turbine operational.

Field Oriented Control (FOC) framework targets to control wind turbines from the side of generator converter, where objective is to maintain its operation speed at rated value. This way, efficient operation of wind turbine is achieved. In addition, the basic idea is to decouple slow dynamics magnetizing flux and fast dynamics torque building elements, such that each of them can be separately controlled with the aim of very fast torque control. This allows one to design standard PID control loops. More details about FOC can be found e.g. in Leonhard (2001).

In this work, we aim to control wind turbine process. Particularly, we will construct MPC strategy for a rotor-flux-based FOC with voltage-controlled converter, where control voltages are subjected to safety criteria. The main challenge behind this work is that the voltage constraints have circular character and the controlled system is fast dynamics, thus maintaining rapid sampling time of 0.4 millisecond is crucial. We show that low complexity MPC policy which tracks desired reference and provides constraint satisfaction can be derived via combining offset free tracking approach with move-blocking technique and by applying polyhedral approximation of circular voltage restrictions.

8.1 Introduction to the Controlled Process

In this section we will introduce the experimental process. Particularly, we will start by the process description, then we will derive a model of the wind turbine and specify considered constraints of the system.

8.1.1 Process Setup

The wind turbine process is part of the laboratory equipment in the Faculty of Electrical Engineering and Computing (FER) of the University of Zagreb, in Croatia and, designed as part of the dissertation thesis (Lešić, 2014). Since, for the purposes of this work, we will provide only an essential information about the experimental setup, we encourage interested readers to find more details in (Lešić, 2014, Appendix C).

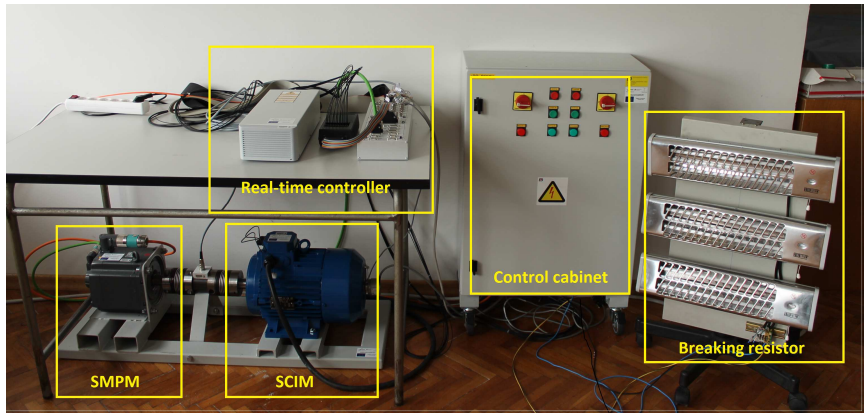


Figure 8.1: Laboratory setup for the wind turbine experiment.

The Laboratory setup of the wind turbine is depicted in the Figure 8.1. The wind turbine is here represented by two main parts. This first one is a squirrel-cage induction machine (SCIM), which represents the actual generator of the energy. The second part is a permanent magnet synchronous machine (PMSM), which acts as an emulator of wind turbine aerodynamical torque. Both parts have coupled shafts. The real-time controller is a dSPACE 1103 with 1 GHz processor and numerous digital and analog input output interfaces (dSPACE, 2016), which directly communicates with MATLAB via dSPACE ControlDesk software package. Next, the control cabinet incorporates all machine connectors, sensors and motor protec-

tion units. The breaking resistor, i.e. heater, is here basically used to consume the generated power energy.

8.1.2 Model of the Wind Turbine Generator

Dynamics of the SCIM, i.e., the wind turbine generator, for the purposes of voltage FOC, are described by two following ODEs:

$$u_{sd}(t) = k_s i_{sd}(t) + L_l \frac{\mathrm{d}i_{sd}(t)}{\mathrm{d}t} - \frac{1}{T_r}\frac{L_m^2}{L_r} i_{mr}(t) - \omega_e(t) L_l i_{sq}(t), \tag{8.1a}$$

$$u_{sq}(t) = R_s i_{sq}(t) + L_l \frac{\mathrm{d}i_{sq}(t)}{\mathrm{d}t} + \omega_e(t)\frac{L_m^2}{L_r} i_{mr}(t) + \omega_e(t) L_l i_{sd}(t), \tag{8.1b}$$

with

$$k_s = R_s + \frac{L_m^2}{L_r^2} R_r,\ L_l = L_s - \frac{L_m^2}{L_r},$$

where variables $i_{sd}(t)$, $i_{sq}(t)$, $u_{sd}(t)$, $u_{sq}(t)$ and $\omega_e(t)$ represent stator direct current, quadrature current, direct voltage, quadrature voltage and electrical angular speed, respectively, and $i_{mr}(t)$ is magnetizing current of squirrel-cage induction machine. Next, constants L_l, L_m, L_r and L_s denote leakage, mutual, rotor and stator inductances, R_s and R_r are stator and rotor resistance, respectively.

The real-time implementation of (8.1) might be jeopardized due to two reasons. The first one is nonlinear nature of (8.1), e.g. multiplication between $\omega_e(t)$ and $i_{sq}(t)$ in (8.1a). The second obstacle is rapid sampling rate of about 0.4 milliseconds, which is crucial to maintain for the proper operation of SCIM. To tackle this issue a linear model is derived next.

To proceed, let us introduce decoupling variables

$$\gamma_1(t) = \frac{1}{T_r}\frac{L_m^2}{L_r} i_{mr}(t) - \omega_e(t) L_l i_{sq}(t), \tag{8.3a}$$

$$\gamma_2(t) = -\omega_e(t)\frac{L_m^2}{L_r} i_{mr}(t) + \omega_e(t) L_l i_{sd}(t), \tag{8.3b}$$

based on which we can rewrite (8.1) into

$$u_{sd}(t) + \gamma_1(t) = k_s i_{sd}(t) + L_l \frac{\mathrm{d}i_{sd}(t)}{\mathrm{d}t}, \tag{8.4a}$$

$$u_{sq}(t) + \gamma_2(t) = R_s i_{sq}(t) + L_l \frac{\mathrm{d}i_{sq}(t)}{\mathrm{d}t}. \tag{8.4b}$$

By neglecting very slow variation of $i_{mr}(t)$ and assuming that at each time instant t the system nonlinearities are ideally compensated, i.e. $\gamma_1(\omega_e(t), i_{sq}(t), i_{mr}(t)) \approx$

constant and $\gamma_2(\omega_e(t), i_{sd}(t), i_{sq}(t), i_{mr}(t)) \approx$ *constant*, then dynamics of the system (8.1) can be defined by the following transfer functions

$$\frac{i_{sd}(t)}{u_{cd}(t)} = \frac{\frac{1}{k_s}}{\frac{L_l}{k_s}s + 1}, \tag{8.5a}$$

$$\frac{i_{sq}(t)}{u_{cd}(t)} = \frac{\frac{1}{R_s}}{\frac{L_l}{R_s}s + 1}, \tag{8.5b}$$

where s is the Laplace operator and control voltages are given by

$$u_{cd}(t) = u_{sd}(t) + \gamma_1(t), \tag{8.6a}$$

$$u_{cq}(t) = u_{sq}(t) + \gamma_2(t). \tag{8.6b}$$

By using the zero-order hold technique to discretize (8.5) with sampling time T_s, we obtain

$$i_{sd}(t+1) = b_1 i_{sd}(t) + d_1 u_{cd}(t), \tag{8.7a}$$

$$i_{sq}(t+1) = b_2 i_{sq}(t) + d_2 u_{cq}(t), \tag{8.7b}$$

where

$$d_1 = \frac{(1-b_1)}{k_s},\ b_1 = e^{-\frac{k_s T_s}{L_l}},$$
$$d_2 = \frac{(1-b_2)}{R_s},\ b_2 = e^{-\frac{R_s T_s}{L_l}}.$$

Finally, the discrete-time state-space model of the wind turbine is represented by

$$x(t+1) = Ax(t) + Bu(t), \tag{8.9a}$$

$$y(t) = Cx(t), \tag{8.9b}$$

where $x(t) = [i_{sd}(t), i_{sq}(t)]^T \in \mathbb{R}^2$ is a state vector, $u(t) = [u_{cd}(t), u_{cq}(t)]^T \in \mathbb{R}^2$ is a control vector, $y(t) = [i_{sd}(t), i_{sq}(t)]^T \in \mathbb{R}^2$ is a output vector and matrices are given as

$$A = \begin{bmatrix} b_1 & 0 \\ 0 & b_2 \end{bmatrix}, \quad B = \begin{bmatrix} d_1 & 0 \\ 0 & d_2 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

8.1.3 Tracking Problem of MPC

Tracking of the reference represents one of the most important property of all feedback controllers, especially when regulating towards origin is no longer sufficient.

In the concept of FOC, tracking of y_{ref} leads to optimal operation of the wind turbine, thus to the efficient energy production. In order to achieve offset free tracking, we essentially need to steer the system (8.9) to the steady state such that the controlled outputs (or states) will attain values of the references, i.e.

$$\begin{bmatrix} A-I & B \\ C & 0 \end{bmatrix} \begin{bmatrix} x^s \\ u^s \end{bmatrix} = \begin{bmatrix} 0 \\ y^s \end{bmatrix}, \tag{8.11}$$

where $y^s = y_{\text{ref}}$. In MPC, this can by done by penalizing the differences $\|y_k - y_{\text{ref}}\|$ and $\|u_k - u^s\|$ see (Muske, 1997; Shead et al., 2010). We need to keep in mind that this approach provides offset free tracking only if the prediction model is identical with the real process. However, in case of plant model mismatch (see Figure 3.2), one needs to consider another technique to remove the tracking error e.g. as disturbance modeling. The main idea of disturbance modeling (Borrelli and Morari, 2007; Muske and Badgwell, 2002; Pannocchia and Rawlings, 2003) is to enhance the system model by disturbance model which predicts the mismatch between predicted and measured output. Subsequently, inaccurate states and outputs of the model are then corrected, what results in the offset free tracking. The model in (8.9) can be augmented as follows

$$x(t+1) = Ax(t) + Bu(t) + B_d d(t), \tag{8.12a}$$

$$d(t+1) = d(t), \tag{8.12b}$$

$$y(t) = Cx(t) + C_d d(t), \tag{8.12c}$$

where B_d and C_d denote disturbance matrices and $d(t) \in \mathbb{R}^2$ is vector of disturbances. For the system in (8.12) the state and disturbance observer is given as:

$$\begin{bmatrix} \hat{x}(t+1) \\ \hat{d}(t+1) \end{bmatrix} = \begin{bmatrix} A & B_d \\ 0 & I \end{bmatrix} \begin{bmatrix} \hat{x}(t) \\ \hat{d}(t) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u(t) + \begin{bmatrix} K_1 \\ K_2 \end{bmatrix} \left(y_{\text{p}}(t) - C\hat{x}(t) - C_d\hat{d}(t) \right), \tag{8.13}$$

where $\hat{x}(t)$, $\hat{d}(t)$ denote estimated values of states and disturbances, y_{p} is the measured (estimated) output from the process, gains K_1 and K_2 are chosen such that the observer is stable and further tuned to achieve desirable dynamic for observer convergence.

Remark 8.1.1 *As reported in (Borrelli et al., 2016, Section* 13.6*) if dimensions of disturbances $d(t)$ and outputs $y(t)$ are equivalent, (C, A) is observable and $det(A - I - B_d C) \neq 0$, then augmented model in* (8.12) *is observable and we can choose*

$C_d = I$. Moreover, if plant has no integrators, i.e. $det(A - I) \neq 0$, we can choose $B_d = 0$.

8.1.4 Constraints

Electrical machines and generators are usually overdimensioned to cope with occasional overshoots of rated values and short overloads during transients that occur in normal operation. This potential problem is usually solved by introducing practical implementation aspects such as ramps applied to references, soft-starters during start-up, or protection units. Solving these problems from the perspective of control algorithms by imposing and respecting constraints may lead to cheaper generator designs and unnecessary shut-down avoidance. Therefore, to keep the generator within rated operation, we consider following constraints on the rated voltage

$$(u_1(t) - \gamma_1(t))^2 + (u_2(t) - \gamma_2(t))^2 \leq \left(\frac{u_{dc}}{\sqrt{3}}\right)^2, \tag{8.14}$$

where u_{dc} is power converter DC-link voltage (for more information about power converter topology and operation, see e.g. (Leonhard, 2001)). Note that if voltage constraints are satisfied, currents are consequently kept below rated values. Since the MPC is formulated as a tracking problem, currents constraints are also entrusted to outer control loop that provides reference trajectories. We need to emphasize, however, two difficulties of the aforementioned constraints. First one is that constraints in (8.14) are circular, what leads to complex optimization problems. This issue is usually tackled by replacing (8.14) with a set of linear constraints, i.e. via (inner/outer) polyhedral approximation. The second problem is that the center of this circle can change its coordinates via decoupling variables γ, what has to be taken into account during the MPC design. This two important facts are illustrated in Figure 8.2.

8.2 Control of the Wind Turbine Generator

This section provides information of the experimental setup from the control point of view. We start by a short description of control scheme and clarifying what is the control objective of this experiment Then we discuss approach which tackles complexity obstacle of the MPC optimization problem. Finally, after we state the

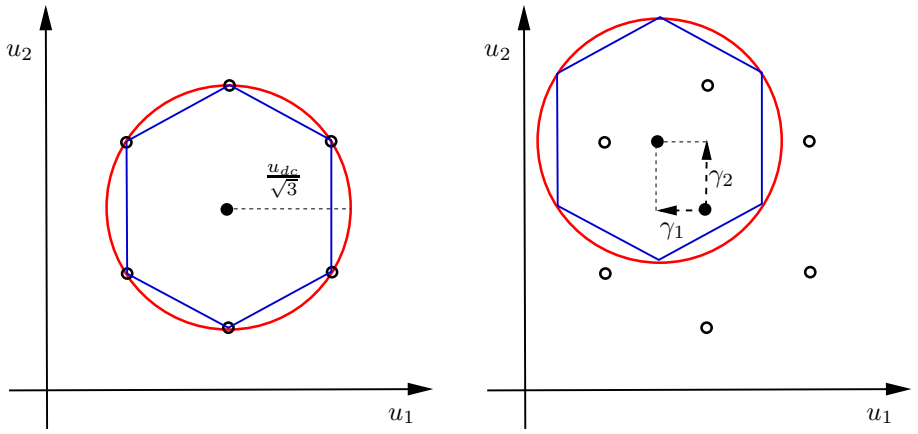


Figure 8.2: Illustration of constraints. Red circle denotes process constraints, blue polytope is inner approximation, dashed-black line represents radius of the circle and the black circle denotes center of constrains that can be shifted by γ.

experimental model and constraints, we introduce used MPC formulation for the field-oriented control of SCIM.

8.2.1 Control Objective

Consider that at each sampling instant we are given the optimal values of $i^\star_{sd}$ (for the most of time attains the rated value $i_{sdn} = 4.8\,\mathrm{A}$) and $i^\star_{sq}$ (given from the outer torque controller). Moreover, assume that Kalman filter (KF) (Lešić, 2014, Chapter 10) is well tuned and embedded in the control loop. Our objective is to design and implement an MPC policy, which will replace decoupled PID controllers, such that constraints in (8.14) will be satisfied, the sampling period $T_s = 4 \cdot 10^{-4}$ will be preserved and given references $y_{\mathrm{ref}} := [i^\star_{sd}, i^\star_{sq}]^T$ will be tracked.

The MPC control scheme of SCIM is shown in Figure 8.3. Here, the optimal control inputs u from the MPC feedback law are corrected by the decoupling parameters $\hat{\gamma}$, what yields the optimal voltages $u^\star_{sd}$ and $u^\star_{sq}$, which are transformed into three phase $u^\star_{abc}$ sine voltages via current angle of the (d,q)-transformation ρ. Finally, $u^\star_{abc}$ is then converted into associated current i_a, i_b, i_c and fed to the generator (SCIM), the consequence of which is the generator mechanical speed ω_g,

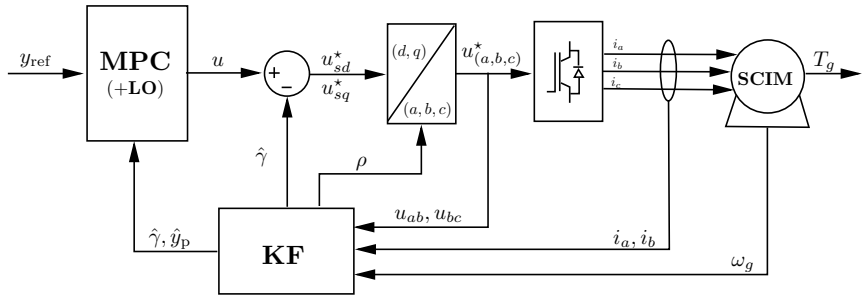


Figure 8.3: Wind turbine generator tracking MPC control scheme.

torque T_g and thus power. Subsequently, KF estimates decoupling parameters $\hat{\gamma}$ and outputs $\hat{y}_{\text{p}}$. Process outputs $\hat{y}_{\text{p}}$ are then given to LO, where states $\hat{x}$ and disturbances $\hat{d}$ are estimated. Finally, based on estimates from LO and references y_{ref} (from outer controller), MPC computes new sequence of optimal control inputs u, what completes the control scheme.

8.2.2 Complexity Problem

It is known (see Chapter 3) that the main implementation obstacle of MPC is the computation burden since the based on the RHC the optimization problem has to be coped within the duration of one sampling period, which in our case is $T_s = 4 \cdot 10^{-4}$ seconds. To even more stress this issue we need to point out that in the control scheme shown in Figure 8.3 we have various signal conversions, as well as KF, which requires majority of the available computation time. Therefore, the online evaluation of MPC feedback law has to be much faster. To achieve this rapid control frequency, even with the modern control platform in hand, it usually takes to employ state-of-the-art optimization solvers. Besides that these optimization solvers are quite expensive and since we had not access to any of those, during our experiments we have been forced to find a different way. The solution for this problem was quite obvious and led to explicit MPC.

As we are already know from the Chapter 4, in explicit MPC we exploit parametric programming to offline precalculate the MPC optimization problem for all feasible initial conditions what yields an explicit optimizer in form of a polytopic PWA function the online evaluation of which can be efficiently handled only by means of simple mathematical operations. Here, on the one side we do not longer

need to employ any expensive optimization solver, but on the other side the requested fast evaluation of control inputs is not yet guaranteed. To see this, let us remind that online execution of explicit MPC feedback laws is essentially restricted only to a point location problem. From Section 4.3.1 we know that there exists multiple algorithms for this task, but all of them are directly related to the number of regions of the explicit solution, thus to its complexity.

In this work, since the complexity of explicit optimizers has grown extremely fast (e.g. for prediction horizon $N = 3$ we obtained explicit solution with more then one thousand of regions), we have decided to decrease the online evaluation of explicit MPC by directly reducing the complexity of the MPC optimization problem. Particularly, by exploiting so-called move-blocking technique.

Move-blocking represents practical and widely used approach in MPC strategy, which is used to reduce the complexity associated with finite-horizon control problems (Qin and Badgewell, 2003; Tøndel and Johansen, 2002). The concept of this technique is to force the control inputs to remain the same along certain prediction steps. In other words, let u_k for $k = 0, \ldots, N-1$ to be the sequence of optimized control moves over the prediction horizon N. Now, denote N_c to be the control horizon, where $N_c < N$, through which move-blocking imposes

$$u_k = u_{k-1},\ k = N_c, \ldots, N. \tag{8.15}$$

This way the degree of freedom is reduced exactly by $N - N_c$ variables. To see this, note that (8.15) introduces $N - N_c$ equality constraints into the MPC optimization problem, that can be subsequently eliminated via technique shown Section 2.2.4. On the other hand, the price which comes with this complexity reduction is that such design control policies do no longer guarantee stability or constraints satisfaction. Yet, as it was shown in (Cagienard et al., 2007b) it is possible to provide theses guarantees by using time-varying scheme called move window blocking.

Remark 8.2.1 *As it was discussed in Section 4.3.1, online evaluation of explicit MPC can be more efficient e.g. by employing binary search tree technique as the algorithm for the point location problem. However, from the side of this experiment, with model and constraints as in* (8.12) *and* (8.20)*, we were not able to construct an efficient tree structure, even though that multiple variations of MPC formulations have been assumed. The acceleration of point location problem was approximately* 20% *(compared to the standard sequential search technique). Yet, this reduction was not sufficient for the real-time application, if MPC had longer prediction horizons.*

Remark 8.2.2 *Another approach how to decrease complexity of explicit controllers is via applying memory reduction techniques e.g. such as is shown in Chapter 5. This direction was not our primal choice since application of those techniques would require additional computation time and majority of the MPC tuning had to be performed directly in the real process.*

8.2.3 Control Components of the Experiment

In Section 8.1.2 we have derived a general LTI model for the wind turbine. Now, by applying parameters of our experimental process into (8.9) and with $T_s = 4 \cdot 10^{-4}$ we obtain matrices

$$A = \begin{bmatrix} 0.9679 & 0 \\ 0 & 0.9816 \end{bmatrix}, \; B = \begin{bmatrix} 0.0294 & 0 \\ 0 & 0.0297 \end{bmatrix}, \; C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{8.16a}$$

Now, since system has no integrator, (C, A) is observable and $det(A - I - B_d C) \neq 0$ holds we can define disturbance model as

$$B_d = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \; C_d = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \tag{8.17a}$$

and therefore the augmented model in (8.12) has following form

$$x(t+1) = \begin{bmatrix} 0.9679 & 0 \\ 0 & 0.9816 \end{bmatrix} x(t) + \begin{bmatrix} 0.0294 & 0 \\ 0 & 0.0297 \end{bmatrix} u(t), \tag{8.18a}$$

$$d(t+1) = d(t), \tag{8.18b}$$

$$y(t) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} d(t). \tag{8.18c}$$

The Luenberger observer (LO) matrices K_1 and K_2 have been chosen such that (8.13) would be stable:

$$K_1 = \begin{bmatrix} 0.6710 & 0 \\ 0 & 0.6801 \end{bmatrix}, \; K_2 = \begin{bmatrix} 0.2979 & 0 \\ 0 & 0.2983 \end{bmatrix}. \tag{8.19}$$

Let us now specify constraints considered in this experiment. In order to provide safe operation mode of the wind turbine we restrict the voltage signal u to attain values only within a circle with radius of 260. To keep complexity low, we have approximated (8.14) by a hexagon what led to following voltage constraints

$$G_c\left(u(t)+\gamma(t)\right) \leq F_c, \tag{8.20}$$

with matrices

$$G_c = \begin{bmatrix} 0 & 1 \\ -1.7321 & 1 \\ -1.7321 & -1 \\ 0 & -1 \\ 1.7321 & -1 \\ 1.7321 & 1 \end{bmatrix}, \; F_c = \begin{bmatrix} 225.1666 \\ 450.3332 \\ 450.3332 \\ 225.1666 \\ 450.3332 \\ 450.3332 \end{bmatrix}.$$

8.2.4 MPC Formulation

Now, with model (8.18) and constants (8.20) in hand, let us define MPC optimization problem as follows

$$\min_{u_0,\ldots,u_{N-1}} \sum_{k=0}^{N-1} \left(||Q_{\mathrm{y}}(y_k - y_{\mathrm{ref}}(t)))||_2^2 + ||Q_{\mathrm{u}}(u_k - u^s(t))||_2^2 \right) \tag{8.22a}$$

$$\text{s.t. } x_{k+1} = Ax_k + Bu_k, \quad k = 0, \ldots, N-1, \tag{8.22b}$$

$$y_k = Cx_k + C_d\hat{d}(t), \tag{8.22c}$$

$$x_0 = \hat{x}(t), \tag{8.22d}$$

$$u_k = u_{k-1}, \quad k = N_c, \ldots, N-1, \tag{8.22e}$$

$$G_c(u_k - \hat{\gamma}(t)) \leq F_c, \quad k = 0, \ldots, N-1, \tag{8.22f}$$

where the steady-state control input $u^s(t)$ at sample time t is given by:

$$\begin{bmatrix} A-I & B \\ C & 0 \end{bmatrix} \begin{bmatrix} \hat{x}^s(t) \\ u^s(t) \end{bmatrix} = \begin{bmatrix} 0 \\ y_{\mathrm{ref}}(t) - C_d\hat{d}(t) \end{bmatrix}. \tag{8.23}$$

Denote $x_k \in \mathbb{R}^2$ and $u_k \in \mathbb{R}^2$ state and input vector, respectively, at prediction step k, $\hat{x}(t) \in \mathbb{R}^2$ and $\hat{d}(t) \in \mathbb{R}^2$ are state and disturbance estimates (from LO), $\hat{\gamma}(t) \in \mathbb{R}^2$ is vector of decoupling voltages (from KF) and $y_{\mathrm{ref}}(t)$ is reference at sample time t. Next, Q_{u} and Q_{y} are positive definite weighting matrices, N denotes prediction horizon and N_c represents control horizon. The term $||\cdot||_2^2$ denotes the squared Euclidean norm, i.e. $||Q_{\mathrm{u}}\left(u_k - u^s(t)\right)||_2^2$ equals to $\left(u_k - u^s(t)\right)^T Q_{\mathrm{u}}\left(u_k - u^s(t)\right)$.

Let us now closer explain optimization problem in (8.22). The first term in the objective function (8.22a) penalizes the distance between the predicted outputs

$y_k = Cx_k + C_d\hat{d}(t)$ and the reference $y_{\text{ref}}(t)$, i.e. the tracking error. The second term, on the other hand, penalizes predicted and steady-state control actions. The constraint in (8.22b) denotes prediction model in (8.18) where (8.18b) is omitted as it is constant during the entire prediction and (8.18c) is directly substituted in optimization problem. The initial constraint (8.22d) enables the recursion from estimated to predicted states, constraint in (8.22e) denotes the move-blocking technique and (8.22f) are the safety constraints as in (8.20).

Since objective function in (8.22a) is quadratic and all constraints (8.22b)-(8.22f) are linear, we have that optimization problem in (8.22) is QP with optimization variables[1] $z(t) := [u_0, \dots, u_{N-1}]^T \in \mathbb{R}^N$ and parameter vector $\theta(t) := [\hat{x}^T(t), \hat{\gamma}(t)^T, (C_d\hat{d}(t) - y_{\text{ref}}(t))^T]^T \in \mathbb{R}^6$. Note that we have decreased the parameter space via merging $y_{\text{ref}}(t)$ and $\hat{d}(t)$ together. We were able to do this, as neither of those values are present in constraints (8.22b)-(8.22f) and since steady-state control input $u^s(t)$ is a function of this parameters, i.e. $u^s(y_{\text{ref}}(t) - C_d\hat{d}(t))$, see (8.23). However, if e.g. states of the system would be considered, which have to be corrected by the disturbances to attain real values, i.e. $G_x(x_k + \hat{d}(t)) \leq F_x$, then this parametrization is not possible.

The MPC optimization problem (8.22) was tuned and the best performance, with the respect of complexity, was obtained with:

$$Q_{\text{y}} = \begin{bmatrix} 500 & 0 \\ 0 & 250 \end{bmatrix}, \; Q_{\text{u}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \; N = 5, \; N_c = 1. \tag{8.24}$$

Solving (8.22) with (8.24) in MPT3, led to explicit solution defined over 33 regions in dimension of 6. This controller was then exported into self-contained function and added to the control scheme depicted in Figure 8.3.

8.3 Experimental Results

In what follows, we will demonstrate experimental results, where two main scenarios are considered. The first one illustrates the tracking performances of the controller and the second one is aimed to verify if the safety control operation of generator is provided, i.e. if constrains in (8.20) are satisfied. Results are also compared with

[1] The number of decision variables was actually $z(t) := [u_0, \dots, u_{N_c-1}]^T \in \mathbb{R}^{N_c}$ as in MPT3, where multiparametric solution was obtained, the constraint elimination technique is used, cf. Section 2.2.4

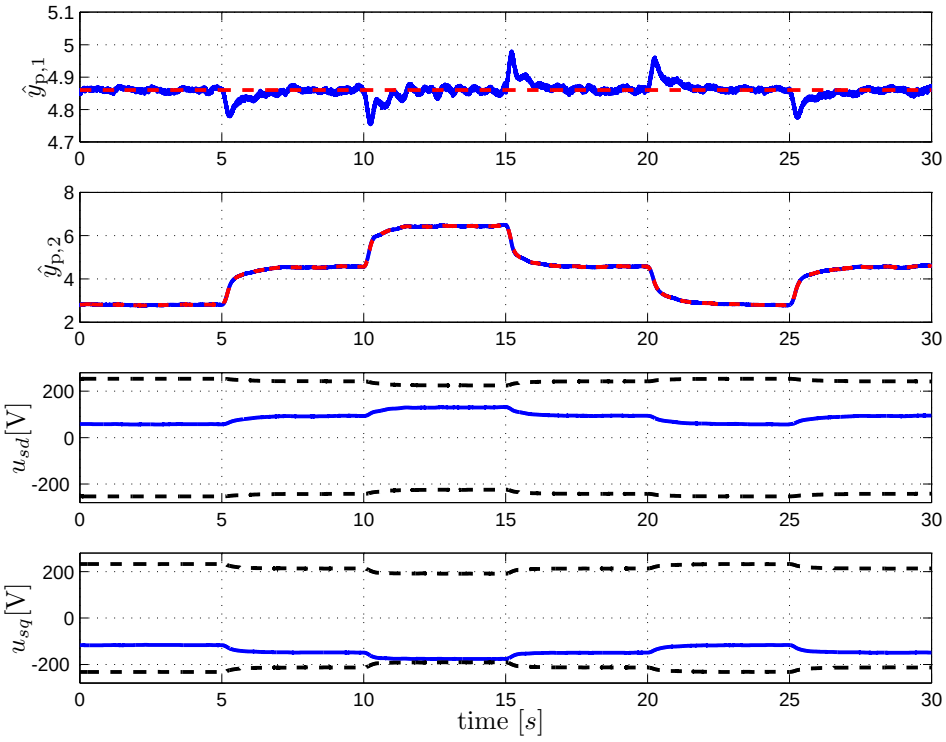


Figure 8.4: Control performance of MPC with respect to no-stochastic wind. Here, references are denoted by red-dashed lines and black-dashed lines are control voltage constraints.

the performance of typically used PID controller. Moreover, we would like to note that in presented graphs we will not show directly control inputs of MPC (PID), but we will instead show the real control voltages as in (8.6), i.e. $u_{sd} = u_1 - \hat{\gamma}_1$ and $u_{sq} = u_2 - \hat{\gamma}_2$.

We would like to recall that generator control is performed in the first control mode of wind turbines, i.e. operation below rated wind speed, and during our experiment i_{sd}^{*} was set to $i_{sd}^{\star} = i_{sdn} = 4.8\,\mathrm{A}$. Particularly, wind turbine was affected by simulated wind speed of $[4, 5, 6, 5, 4, 5]\,\mathrm{m\,s^{-1}}$ that was changing with frequency of 5 seconds. Figure 8.4 illustrates tracking performance of MPC. It may be observed that while the first output $\hat{y}_{\mathrm{p},1}$ (stator direct current) had to track the constant (rated) value i_{sdn}, the second state $\hat{y}_{\mathrm{p},2}$ (quadrature current) had to follow the reference $i_{sq}^{\star}$ that was given by the outer torque controller. In the graph we can

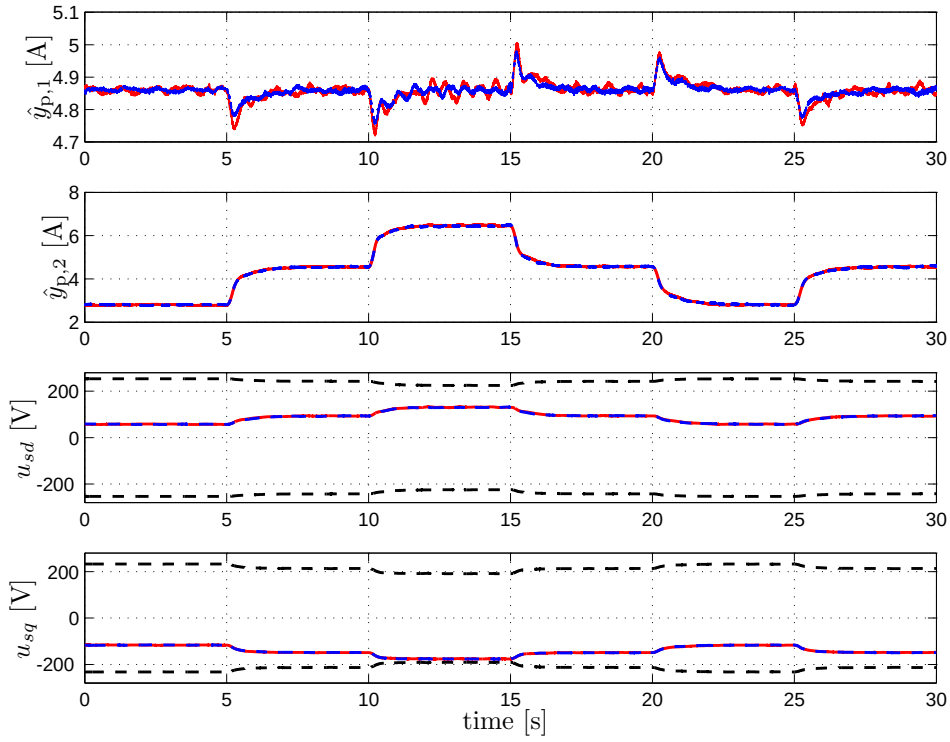


Figure 8.5: Comparison of MPC (blue-dashed lines) and PID (red lines) control performances. Black-dashed lines denote control voltage constraints.

observe that MPC has indeed achieved this goal. Moreover, offset free tracking was provided due to disturbance modeling described in Section 8.1.3. Note that tracking of the first reference is much harder to obtain than the second one. This had an impact on selection of weighting matrix Q_{y}. Next, we can see weak oscillating behavior during the strongest wind speed within interval of 10-15 seconds. This occurs due to edge operation between the two operating modes, where blade pitching is occasionally taking over the wind turbine control.

Subsequently, with the same control setup, we have controlled process with decoupled PID controllers and their comparison with MPC policy is depicted in Figure 8.5.Here, the blue trajectories denote MPC policy performance and red ones PID controllers. By comparing these two strategies we can come to conclusion that both MPC and PID have good control performances. Moreover, in this case, MPC was able to better compensate overshoots and undershoot of $\hat{y}_{\mathrm{p},1}$, but, the price

which we have paid for it was worse tracking of $y_{\mathrm{ref},2}$. To be more exact, based on ISE indicator, the tracking performance of MPC was 41% better w.r.t. $\hat{y}_{\mathrm{p},1}$, but by 4% worse w.r.t. $\hat{y}_{\mathrm{p},2}$, when compared with PID. This is a direct consequence of the weighting matrices in (8.24), where we have selected greater penalization on the first controlled output.

In Figure 8.6, control performance of MPC was also evaluated with respect to stochastic wind. Here, we can observe that MPC has also provided good control performance in more dynamic and realistic operation.

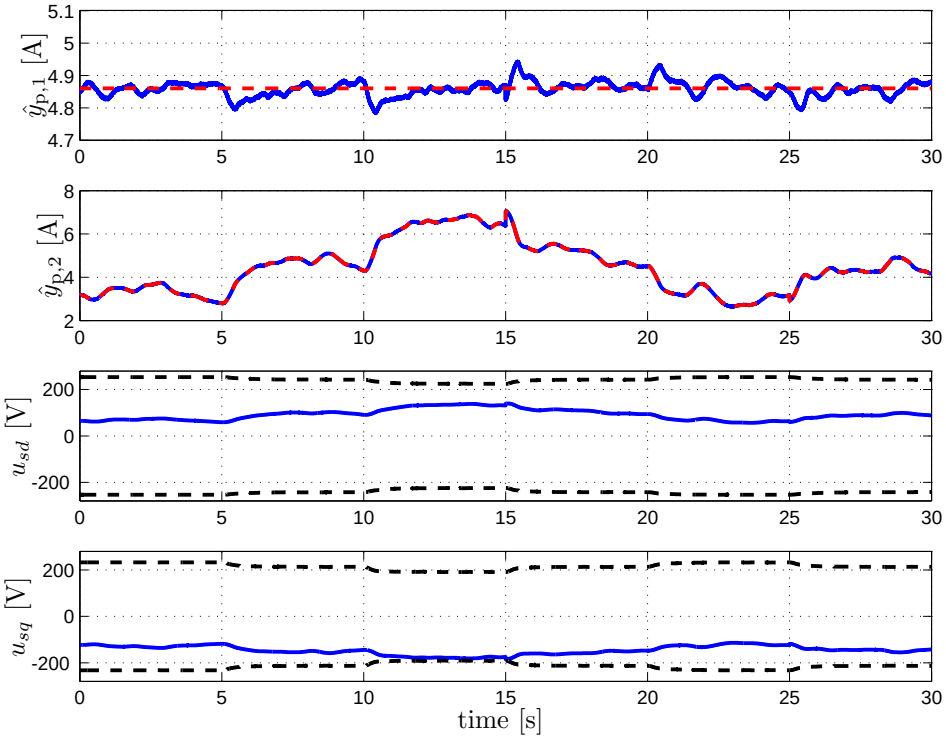


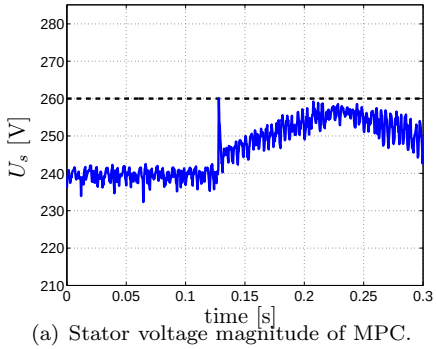
Figure 8.6: MPC control performance with respect to stochastic wind. References are denoted by red-dashed lines and control voltage constraints by black-dashed lines.

Finally, second scenario of this experiment is shown in Figure 8.7, where violation of constraints in (8.20) is evaluated. For illustrative purposes, we have used value of $U_s(t) = \sqrt{(u_1(t) - \hat{\gamma}_1(t))^2 + (u_2(t) - \hat{\gamma}_2(t))^2} \leq \frac{u_{dc}}{\sqrt{3}}$, which represents stator voltage magnitude that has to be kept within specified radius $\frac{u_{dc}}{\sqrt{3}}$ (see (8.14)).

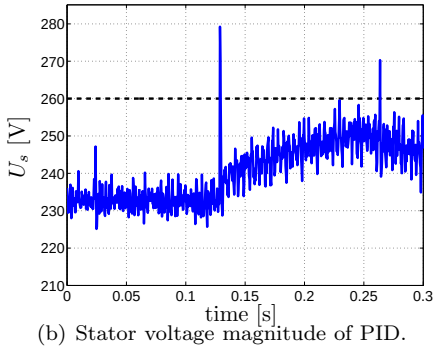
Here, to demonstrate constraints satisfaction, we have done one short step change. The difference, however, was that this step did not go the outer torque controller (which essentially 'filtered' this signal due to wind turbine moment of inertia and slow system dynamics, as we have seen in previous figures), but directly to the generator. In other words, we have performed step change on the second reference $y_{\text{ref},2} = i^{\star}_{sq}$. This scenario rarely occurs in wind turbine applications (basically only during start up at cut-in speed) but is very common in other machine applications such as servo drives, motion control etc. Here we used it to point out constrained control outputs. From the presented graphs one can observe that aforementioned step had an immediate response from the side of controllers. However, while PID controllers have not respected control voltage constraints, MPC has provided constraints satisfaction due to (8.20) embedded in the optimization problem (8.22). For convenience, we have also shown tracking performances of MPC and PI, which are almost the same. Notice, however, that the second state converges slower in case of MPC policy. This is caused due to the weighting matrices in (8.22a), where the second output offset is less penalized (8.24). We remind that they have been tuned with respect to the outer torque controller, which shapes (smoother) references to MPC. Therefore, as we have seen in previous figures, it i more important to penalize the first output offset. Moreover, performance of LO can be observed in this figure. We can see that LO tracks $\hat{y}_{\text{p}}$ well, thus provides to the MPC policy relevant estimates of both model states $\hat{x}$ and disturbances $\hat{d}$.

8.4 Conclusions

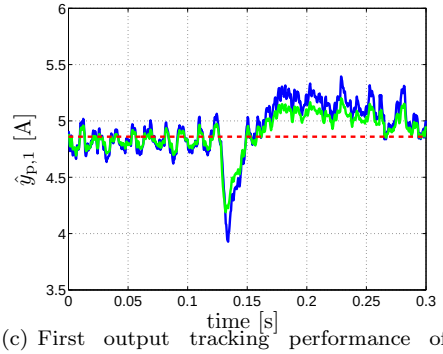
In this chapter, we have proposed MPC strategy to wind turbine generator. To provide good control performance we used several control concepts such as move-blocking, disturbance modeling and parametric programming. Based on presented figures we have seen that MPC has adopted to stochastic nature of wind, exhibited offset free tracking and provided constraints satisfaction. On the other hand, even thought that commonly used PID controllers have the same control performance, they did not meet constraints. Therefore, we can conclude that proposed MPC policy represents better option how to provide additional protection to the generator, what prolongs life cycle of the wind turbine and thus leads to greater profits. We would like also to emphasize that we have achieved this without changing control setup, i.e. optimization solver or more powerful control platform is not required.



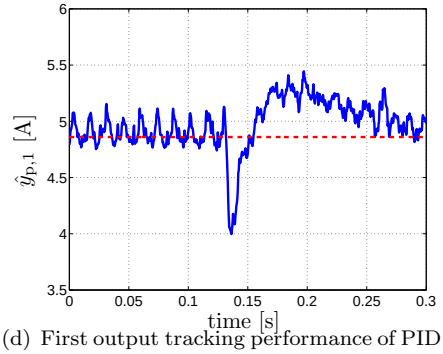
(a) Stator voltage magnitude of MPC.



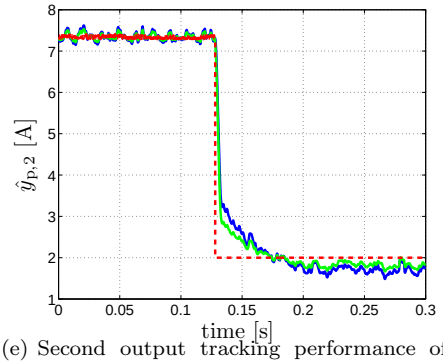
(b) Stator voltage magnitude of PID.



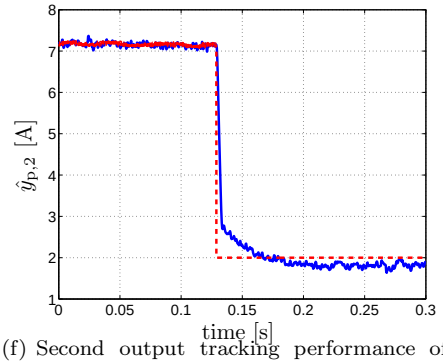
(c) First output tracking performance of MPC.



(d) First output tracking performance of PID.



(e) Second output tracking performance of MPC.



(f) Second output tracking performance of PID.

Figure 8.7: Verification of constraints satisfaction. Step change was performed directly on the second reference $y_{\text{ref},2}$. Here, black-dashed lines denote constraints, red-dashed lines denote references and blue lines represent estimated values of KF. Moreover, green lines denote estimated output of LO, i.e. $\hat{x} + \hat{d}$.

Chapter 9

Conclusions and Future Research

This thesis elaborated possibilities of MPC with the target of its fast and memory-efficient implementation. Particularly, we have suggested two principal directions with the same goal, yet with different approach, how to decrease complexity of MPC feedback laws in order to ensure its successful implementation even into low-level control platforms. Moreover, two real applications of MPC have been concerned in this thesis.

In Chapter 5, we have addressed the problem of reducing complexity of explicit MPC controllers. Particularly, we have introduced a novel memory reduction technique that is based on replacing the complex explicit feedback law by its simpler, yet suboptimal, representation. We have shown that this substitution can be achieved via two step procedure. In the first step, we have suggested that new polytopic partition can be obtained by solving the same optimization problem, as the complex explicit solution was constructed, but with shorter prediction horizon. Even though, that this already led to a less complex explicit solution, we were forced to discard it as it exhibited a great performance deterioration, compared to the original controller. Only its polytopic partition was preserved. In the second step, we have suggested how one can significantly reduce the amount of the suboptimality. Specifically, quadratic optimization problem was proposed to optimize new local affine expressions such that error between the approximated and the complex optimizer was minimized, while recursive feasibility and closed-loop stability were provided.

In Chapter 6.2, we have addressed the problem of verifying the performance of MPC polices. Two different techniques were proposed. Their objective is to provide a rigorous answer if all necessary control properties, that were not directly enforced in the MPC formulation, are guaranteed. This way we are able to certificate these simplified feedback controllers, thus to determine if their implementation is safe.

The first proposed approach investigated properties of explicit MPC policies that were exposed to rounding-based quantization effect. This problem was addressed by a three step procedure. Firstly, we have shown how an a-posteriori quantized MPC strategy can be devised, based on the knowledge of the given quantizer and a PWA feedback law, via employing the technique of Voronoi diagram. Secondly, construction of three types of bounding functions have been proposed, as to reflect standard control properties. Namely, recursive feasibility, closed-loop stability and bounded deterioration of performance. Finally, we have defined a rigorous certificate, that verified whether a-posteriori quantized explicit controller was contained within constructed performance boundaries. Subsequently, if a positive answer was obtained, then we had that examined PWA explicit optimizer provided the considered control property, even under a finite precision arithmetic.

The second approach was devoted to safety verification of implicitly defined MPC policies. Specifically, we have proposed non-conservative methods which allow one to verify whether a closed-loop system, composed of a linear controlled plant and an MPC controller, avoids a certain set of unsafe states. The procedure was based on exploiting the Karush-Kuhn-Tucker conditions, that characterized the optimal control inputs. Three principal formulations were presented, each of which led to mixed-integer feasibility problem, the solution of which then indicated if the MPC policy exhibits all necessary safety properties. It was shown that under a minor assumption (existence of a positive invariant terminal set) the verification task can be answered *ad infinitum*, i.e., for an infinite number of time steps. Moreover, we have pointed out that the proposed verification can be also employed to reduce the memory consumption of explicit MPC controllers.

The third part of this thesis was devoted to the implementation of MPC strategy to real systems, where the main objective was to include safety constraints directly to the control algorithm. Two different applications of MPC policy have been proposed.

The pH control problem was addressed in Chapter 7. Here, PID controller was designed to track the desired pH reference in the chemical vessel, where chemical

reaction of acetic acid and the sodium hydroxide has taken place. Subsequently, MPC reference governor technique was proposed in order to improve the control performance of PID. It was shown that in such setup, MPC strategy is able to shape references for the PID controller to compensate overshoots and undershoots of the controlled output, while providing satisfaction of constraints.

In Chapter 8 wind turbine system has been controlled. The objective here was to design fast MPC policy for FOC, which would take into account circular safety input constraints and will track the reference. To achieve this goal, we have exploited several control techniques. The disturbance modeling was introduced to attain offset free reference tracking. In order to mitigate the complexity of MPC policy and provide good control performance, move-blocking technique was employed. Finally, circular constraints were transformed into set of linear inequalities via inner approximated and explicit MPC was used to accelerate computation of optimal control inputs. We have shown, that superiority of MPC, compared to PID, was in satisfying aforementioned safety constraints.

Future Research

Nowadays, besides economical and environmental aspects, safety plays a crucial role in control design. This, however, leads to more complex formulations of MPC what induce the need of applying more expensive control platforms and last, but not least state-of-the-art solvers. This motivates us to improve our proposed methods and control approaches even further. Motivation for future work can be summarized in following points:

- *Memory reduction in explicit MPC:* Proposed memory reduction technique in Section 5 has proved to be a powerful tool how one can decrease complexity of explicit controllers. This motivates us to exploit this technique even further. We would like to merge it with different approaches to decrease memory footprint of these controllers even more, e.g. modify it such that it will be able to reduce memory of region-free explicit solutions, which we have proposed in Kvasnica et al. (2015). Moreover, in Section 6.2 we have shown that proposed verification technique can be used to reduce complexity of explicit solutions. However, the main potential of this approach was not exploited, since this method do not a-priori needs explicit solution in hand. We therefore suggest to employ it directly in the explicit algorithms, such that we will

be able to reduce memory footprint directly when explicit solution is being constructed.

- *Certification techniques:* In verification technique of Section 6.1 we have shown how to certify a-posteriori quantized MPC policy, by means of determining whether quantized PWA feedback lays inside of PWA boundaries. We would like to extend this method for even for richer types of bounding functions, e.g. polynomial boundaries. Next, for the technique in Section 6.2 we will aim at extending it to a richer class of systems, in particular to hybrid linear systems. We will also investigate the inclusion of measured and unmeasured disturbances and parametric uncertainties.

- *Practical implementation of MPC:* In our future research we are also interested to improve the control quality of both real applications. From the pH process control perspective in Section 7, we would like to export proposed MPC policy and control the systems via PLC. To achieve this, we can exploit complexity reduction techniques that are proposed in this work. Moreover, we would like also to elaborate other possibilities, how to deal with the strong non-linearity. One direction how to tackle this issue was proposed e.g. in King (2010). On the other hand, the control of wind turbine in Section 8 has quite straight forward continuation. We would like to follow up the cooperation between universities and improve our control performance via considering so-called fault-tolerant control. Here, additional safety constrains are added to the control policy in order to prevent further structural loads of the wind turbine.

- *Other implementation directions of MPC:* Computer-based traffic fluency control has been addressed in Kalúz et al. (2016). Particularly, a set of cars was controlled as one distributed system. In this scenario the upper-level control system is able to actively communicate with each vehicle, collect the operational data from it, and provide it with the information how to behave. In our future work, we would like to control of each vehicle as the autonomous system.The challenge behind this approach is to design fast and memory-efficient MPC policy which will be able to implement into microcontroller with available memory of less than 2 KBs.

Bibliography

N. N. Anh. *Explicit robust constrained control for linear systems: analysis, implementation and design based on optimization.* PhD thesis, CentraleSupélec, Université Paris Sud, 2015. 104

D.L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook. *The Traveling Salesman Problem: A Computational Study.* Princeton University Press, 2006. 51

E. Asarin, T. Dang, and O. Maler. The d/dt tool for verification of hybrid systems. In *Computer Aided Verification*, pages 365–370. Springer, 2002. 167

F. Aurenhammer. Voronoi diagrams – survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3):345–405, 1991. ISSN 0360-0300. doi: http://doi.acm.org/10.1145/116873.116880. 148

V. Baldoni, N. Berline, J.A. De Loera, M. Köppe, and M. Vergne. How to integrate a polynomial over a simplex. *Mathematics of Computation*, 80(273):297, 2010. 114, 120, 124

M. Baotić. An Efficient Algorithm for Multiparametric Quadratic Programming. Technical report, April 2002. 87, 92, 93, 95, 96

M. Baotić, F. J. Christophersen, and M. Morari. Constrained Optimal Control of Hybrid Systems with a Linear Performance Index. 51(12):1903–1919, December 2006. 74

M. Baotić, F. Borrelli, A. Bemporad, and M. Morari. Efficient on-line computation of constrained optimal control. *SIAM J. Control Optim.*, 47(5):2470–2489,

September 2008. ISSN 0363-0129. doi: 10.1137/060659314. URL http://dx.doi.org/10.1137/060659314. 104, 111

A. Bemporad. Reference governor for constrained nonlinear systems. *Automatic Control, IEEE Transactions on*, 43(3):415–419, 1998. ISSN 0018-9286. doi: 10.1109/9.661611. 198

A. Bemporad. *Hybrid Toolbox - User's Guide*, 2003. 129

A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. 35(3):407–427, March 1999. 49

A. Bemporad, F. D. Torrisi, and M. Morari. Optimization-Based Verification and Stability Characterization of Piecewise Affine and Hybrid Systems. volume 1790, pages 45–58, Pittsburgh, USA, March 2000a. Springer-Verlag. 167

A. Bemporad, F.D. Torrisi, and M. Morari. Performance analysis of piecewise linear systems and model predictive control systems. Sydney, Australia, December 2000b. 167

A. Bemporad, F. Borrelli, and M. Morari. Model Predictive Control Based on Linear Programming—The Explicit Solution. 47(12):1974–1985, December 2002a. 74

A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. 38(1):3–20, January 2002b. 73, 87, 95, 97, 104, 167

A. Bemporad, F. Borrelli, and M. Morari. Min-Max Control of Constrained Uncertain Discrete-Time Linear Systems. 48(9):1600–1606, September 2003. 74

A. Bemporad, A. Oliveri, T. Poggi, and M. Storace. Ultra-fast stabilizing model predictive control via canonical piecewise affine approximations. *Automatic Control, IEEE Transactions on*, 56(12):2883–2897, 2011. ISSN 0018-9286. doi: 10.1109/TAC.2011.2141410. 112

F. Blanchini and S. Miani. *Set-Theoretic Methods in Control*. Birkhauser Boston, 2008. ISBN 978-0-8176-4606-6. 126, 127, 154

V. Bobál, J. Böhm, J. Fessl, and J. Machácek. *Digital Self-tuning Controllers: Algorithms, Implementation and Applications*. Advanced Textbooks in Control and Signal Processing. Springer London, 2006. ISBN 9781846280412. 203

F. Borrelli. *Constrained Optimal Control of Linear and Hybrid Systems*, volume 290. Springer-Verlag, 2003a. 151, 154

F. Borrelli. *Constrained Optimal Control Of Linear And Hybrid Systems*, volume 290 of *Lecture Notes in Control and Information Sciences*. Springer, 2003b. 74, 75, 110, 153, 167

F. Borrelli and M. Morari. Offset free model predictive control. In *Proc. IEEE Conference on Decision and Control*, pages 1245–1250, 2007. 218

F. Borrelli, M. Baotić, A. Bemporad, and M. Morari. Efficient On-Line Computation of Explicit Model Predictive Control. Technical report, August 2001. 102, 104

F. Borrelli, P. Falcone, J. Pekar, and G. Stewart. Reference governor for constrained piecewise affine systems. *Journal of Process Control*, 19(8):1229 – 1237, 2009. ISSN 0959-1524. 198

F. Borrelli, M. Baotić, J. Pekar, and G. Stewart. On the computation of linear model predictive control laws. *Automatica*, 46(6):1035 – 1041, 2010. ISSN 0005-1098. doi: http://dx.doi.org/10.1016/j.automatica.2010.02.031. URL http://www.sciencedirect.com/science/article/pii/S0005109810001032. 111

F. Borrelli, A. Bemporad, and M. Morari. *Predictive Control*. 2016. URL http://www.mpc.berkeley.edu/mpc-course-material. 42, 49, 81, 85, 123, 218

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521833787. 36, 78, 81, 128, 167, 175

T. A. Johansen C. Feller. Explicit mpc of higher-order linear processes via combinatorial multi-parametric quadratic programming. *12th European Control Conference*, 2013. 87, 94

R. Cagienard, P. Grieder, E. Kerrigan, and M. Morari. Move Blocking Strategies in Receding Horizon Control. *Journal of Process Control*, 17(6):563–570, July 2007a. 112

R. Cagienard, P. Grieder, E.C. Kerrigan, and M. Morari. Move blocking strategies in receding horizon control. *Journal of Process Control*, 17(6):563–570, 2007b. 222

F. Christophersen, M. Kvasnica, C. Jones, and M. Morari. Efficient evaluation of piecewise control laws defined over a large number of polyhedra. In *Proceedings of the European Control Conference*, 2007. 104

C.R. Cutler and B.L. Ramaker. Dynamic matrix control—a computer control algorithm. *AICHE national meeting, Houston, TX.*, 1979. 53

C.R. Cutler and B.L. Ramaker. Dynamic matrix control—a computer control algorithm. *In Proceedings of the joint automatic control conference.*, 1980. 53

S. L. de Oliveira Kothare and M. Morari. Contractive model predictive control for constrained nonlinear systems. *IEEE Transactions on Automatic Control*, 45(6): 1053–1071, Jun 2000. ISSN 0018-9286. doi: 10.1109/9.863592. 71

L. Desborough and R. Miller. Increasing customer value of industrial control performance monitoring-honeywell's experience. *AIChE symposium series*, pages 169–189, 2002. 203

S. Di Cairano, H. Park, and I. Kolmanovsky. Model Predictive Control approach for guidance of spacecraft rendezvous and proximity maneuvering. *Int. J. Robust Nonlinear Control*, 22(12):1398–1427, 2012. 74

I. Drca. Nonlinear model predictive control of the four tank process, 2007. 183

dSPACE. Ds1103 ppc controller board, 2016. URL https://www.dspace.com/shared/data/pdf/2014/DS1103.pdf. 215

V. Dua and E. N. Pistikopoulos. An algorithm for the solution of multiparametric mixed integer linear programming problems. 99:123–139, 2000. 74

F. Fagnani and S. Zampieri. Stability analysis and synthesis for scalar linear systems with a quantized feedback. *IEEE Transactions on Automatic Control*, 48 (9):1569–1584, 2003. 142

M. Fu and L. Xie. Finite-Level Quantized Feedback Control for Linear Systems. *Automatic Control, IEEE Transactions on*, 54(5):1165–1170, 2009. ISSN 0018-9286. doi: 10.1109/TAC.2009.2017815. 142

K. Fukuda. Frequently asked questions in polyhedral computation., 2004. URL http://www.ifor.math.ethz.ch/staff/fukuda/polyfaq/polyfaq.html. 38, 42

T. Gal and J. Nedoma. Multiparametric linear programming. *Management Science*, 18:406–442, 1972. 74, 167

A. Gallini. Affine function. from mathworld–a wolfram web resource, created by eric w. weisstein., 2014. URL http://mathworld.wolfram.com/AffineFunction.html. 38

N.I. Georgiev, R. Bryaskova, R. Tzoneva, I. Ugrinova, Ch. Detrembleur, S. Miloshev, A.M. Asiri, A.H. Qusti, and V.B. Bojinov. A novel ph sensitive water soluble fluorescent nanomicellar sensor for potential biomedical applications. *Bioorganic & Medicinal Chemistry*, 21(21):6292 – 6302, 2013. ISSN 0968-0896. doi: http://dx.doi.org/10.1016/j.bmc.2013.08.064. URL http://www.sciencedirect.com/science/article/pii/S0968089613007670. 197

T. Geyer, F. D. Torrisi, and M. Morari. Optimal Complexity Reduction of Piecewise Affine Models Based on Hyperplane Arrangements. pages 1190–1195, Boston, Massachusetts, USA, June 2004. 111

E.G. Gilbert and K.T. Tan. Linear systems with state and control constraints, the theory and application of maximal output admissible sets. *IEEE Transaction on Automatic Control*, 36:1008–1020, 1991. 117

G.C. Goodwin and D.E. Quevedo. Finite alphabet control and estimation. *Internation Journal of Control Automation and Systems*, 1:412–430, 2003. 142

I.S. Gradshteyn and I.M. Ryzhik. *Table of integrals, series, and products*. Elsevier/Academic Press, Amsterdam, seventh edition, 2007. ISBN 978-0-12-373637-6; 0-12-373637-4. Translated from the Russian, Translation edited and with a preface by Alan Jeffrey and Daniel Zwillinger, With one CD-ROM (Windows, Macintosh and UNIX). 62

P. Grieder. *Efficient Computation of Feedback Controllers for Constrained Systems*. PhD thesis, Diss. ETH No. 15785, ETH Zürich, 2004. 75, 110

P. Grieder, M. Kvasnica, M. Baotić, and M. Morari. Stabilizing low complexity feedback control of constrained piecewise affine systems. *Automatica*, 41(10):1683

– 1694, 2005. ISSN 0005-1098. doi: http://dx.doi.org/10.1016/j.automatica.2005.04.016. URL http://www.sciencedirect.com/science/article/pii/S0005109805001482. 172

B. Grunbaum. *Convex Polytopes.* 2000. 38

A. Gupta, S. Bhartiya, and P. S. V. Nataraj. A novel approach to multiparametric quadratic programming. *Automatica*, 47(9):2112–2117, September 2011. ISSN 0005-1098. doi: 10.1016/j.automatica.2011.06.019. 87, 88, 89, 90, 92, 93, 153

Inc. Gurobi Optimization. Gurobi optimizer reference manual, 2014. 52, 54, 129

Global Wind Energy Council GWEC. Global wind statiscits, 2016. URL http://www.gwec.net/wp-content/uploads/vip/GWEC-PRstats-2015_LR_corrected.pdf. 214

T. Henzinger, P. Ho, and H. Wong-Toi. HyTech: A model checker for hybrid systems. In *Computer aided verification*, pages 460–463. Springer, 1997. 167

M. Herceg, M. Kvasnica, C.N. Jones, and M. Morari. Multi-Parametric Toolbox 3.0. In *Proc. of the European Control Conference*, pages 502–510, Zürich, Switzerland, July 17–19 2013a. http://control.ee.ethz.ch/~mpt. 38, 87, 129, 152

M. Herceg, S. Mariéthoz, and M. Morari. Evaluation of piecewise affine control law via graph traversal. In *European Control Conference*, pages 3083–3088, Zurich, Switzerland, July 2013b. 104

J. Holaza, B. Takács, and M. Kvasnica. *Selected Topics in Modelling and Control*, chapter Complexity Reduction in Explicit MPC via Bounded PWA Approximations of the Cost Function, pages 27–32. Number 8. Slovak University of Technology Press, 2012. URL http://www.kirp.chtf.stuba.sk/publication_info.php?id_pub=1356. 112

J. Holaza, B. Takács, and M. Kvasnica. Synthesis of Simple Explicit MPC Optimizers by Function Approximation. In *Proceedings of the 19th International Conference on Process Control*, pages 377–382, Štrbské Pleso, Slovakia, June 18-21 2013. 124, 135

J. Holaza, B. Takács, M. Kvasnica, and S. Di Cairano. Nearly optimal simple explicit mpc controllers with stability and feasibility guarantees. *Optimal Control*

Applications and Methods, 35(6), 28 JUL 2014 2015. doi: 10.1002/oca.2131. URL http://www.kirp.chtf.stuba.sk/publication_info.php?id_pub=1565. 126

B. Houska, H.J. Ferreau, and M. Diehl. ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization. *Optimal Control Applications and Methods*, 32(3):298–312, 2011. 54

G. Huijun and C. Tongwen. A new approach to quantized feedback control systems. *Automatica*, 44(2):534 – 542, 2008. ISSN 0005-1098. doi: http://dx.doi.org/10.1016/j.automatica.2007.06.015. 142

R. Ibrahim. *Practical modelling and control implementation studies on a pH neutralization process pilot plant*. PhD thesis, University of Glasgow, 2008. 197

ILOG, Inc. *CPLEX 8.0 User Manual*. Gentilly Cedex, France, 2003. 52, 54

T. A. Johansen and A. Grancharova. Approximate explicit constrained linear model predictive control via orthogonal search tree. *IEEE Trans. Automat. Contr.*, 48 (5):810–815, 2003. 104, 112

K.E. Johnson. *Adaptive torque control of variable speed wind turbines*. Citeseer, 2004. 214

C.N. Jones and M. Morari. Polytopic approximation of explicit model predictive controllers. *IEEE Trans. Automat. Contr.*, 55(11):2542–2553, 2010. 112

M. Kalúz, J. Holaza, F. Janeček, S. Blažek, and M. Kvasnica. A robotic traffic simulator for teaching of advanced control methods. 2016. URL http://www.kirp.chtf.stuba.sk/publication_info.php?id_pub=1697. 234

S.S. Keerthi and E.G. Gilbert. Optimal Infinite-Horizon Feedback Laws for a General Class of Constrained Discrete-Time Systems: Stability and Moving-Horizon Approximations. *Journal of Optimization Theory and Applications*, 57(2):265–293, May 1988. 70, 71

M. King. *Process Control: A Practical Approach*. Wiley, 2010. ISBN 9780470976661. URL http://books.google.sk/books?id=YKud-kwsA-4C. 234

M. Klaučo, S. Blažek, M. Kvasnica, and M. Fikar. Mixed-integer socp formulation of the path planning problem for heterogeneous multi-vehicle systems. In *Euro-*

pean Control Conference 2014, pages 1474–1479, Strasbourg, France, 2014. URL http://www.kirp.chtf.stuba.sk/publication_info.php?id_pub=1499. 52

M. Kvasnica. *Real-Time Model Predictive Control via Multi-Parametric Programming: Theory and Tools*. VDM Verlag, Saarbruecken, January 2009. 143

M. Kvasnica and M. Fikar. Clipping-based complexity reduction in explicit mpc. *IEEE Transactions on Automatic Control*, 57(7):1878–1883, 2012. 111

M. Kvasnica, J. Löfberg, and M. Fikar. Stabilizing polynomial approximation of explicit MPC. *Automatica*, 47(10):2292–2297, 2011. 112

M. Kvasnica, R. Gondhalekar, A. Szűcs, and M. Fikar. Stabilizing refinement of low-complexity mpc controllers. In *Preprints of 4th IFAC Nonlinear Model Predictive Control Conference*, pages 400–405, 2012. 154

M. Kvasnica, B. Takács, J. Holaza, and S. Di Cairano. On region-free explicit model predictive control. In *54rd IEEE Conference on Decision and Control*, volume 54, pages 3669–3674, Osaka, Japan, December 15-18, 2015 2015. URL http://www.kirp.chtf.stuba.sk/publication_info.php?id_pub=1675. 111, 233

J.B. Lasserre and K.E. Avrachenkov. The Multi-Dimensional Version of $\int_a^b x^p dx$. *The American Mathematical Monthly*, 108(2):151–154, 2001. 120

M. Lazar, D. Munoz de la Pena, W.P.M.H. Heemels, and T. Alamo. On input-to-state stability of min-max nonlinear model predictive control. *Systems & Control Letters*, 57:39–48, 2008. doi: 10.1016/j.sysconle.2007.06.013. 127, 154

W. Leonhard. *Control of electrical drives*. Springer Science & Business Media, 2001. 214, 219

V. Lešić. *Fault-Tolerant Control of a Wind Turbine Subject to Generator Electromechanical Faults*. PhD thesis, FER, University of Zagreb, 2014. 215, 220

J. T. Linderoth and T. K. Ralphs. Noncommercial software for mixed-integer linear programming, 2004. URL http://www.optimization-online.org/DB_HTML/2004/12/1028.html. 52

J. Löfberg. YALMIP : A Toolbox for Modeling and Optimization in MATLAB. In *Proc. of the CACSD Conference*, Taipei, Taiwan, 2004. 44, 66, 129, 186

L. Lu, W. Heemels, and A. Bemporad. Synthesis of low-complexity stabilizing piecewise affine controllers: A control-Lyapunov function approach. In *Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, pages 1227–1232. IEEE, 2011. 112

J.M. Maciejowski. *Predictive Control with Constraints*. Prentice-Hall, 2001. 54

D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. 36(6):789–814, June 2000. 57, 71, 172

D.Q. Mayne and H. Michalska. Receding horizon control of nonlinear systems. *IEEE Transactions on Automatic Control*, 35(7):814–824, Jul 1990. ISSN 0018-9286. doi: 10.1109/9.57020. 70

J. Misiewicz, S. Afonin, and A.S. Ulrich. Control and role of ph in peptide–lipid interactions in oriented membrane samples. *Biochimica et Biophysica Acta (BBA) - Biomembranes*, 1848(3):833 – 841, 2015. ISSN 0005-2736. doi: http://dx.doi.org/10.1016/j.bbamem.2014.12.006. URL http://www.sciencedirect.com/science/article/pii/S0005273614004349. 197

K.R. Muske. Steady-state target optimization in linear model predictive control. In *American Control Conference, 1997. Proceedings of the 1997*, volume 6, pages 3597–3601 vol.6, Jun 1997. doi: 10.1109/ACC.1997.609493. 218

K.R. Muske and T.A. Badgwell. Disturbance modeling for offset-free linear model predictive control. *Journal of Process Control*, 12(5):617 – 632, 2002. ISSN 0959-1524. doi: http://dx.doi.org/10.1016/S0959-1524(01)00051-8. URL http://www.sciencedirect.com/science/article/pii/S0959152401000518. 218

G. Nair, F. Fagnani, S. Zampieri, and R. Evans. Feedback control under data rate constraints: An overview. *Proceedings of the IEEE*, 95(1):108–137, 2007. 142

NEOS. University of wisconsin-madison, 2014. URL http://neos-guide.org/. 35

A. O'Dwyer. *Handbook of PI and PID controller tuning rules*, volume 57. World Scientific, 2009. 204

J. Oravec and M. Bakošová. Piddesign–software for pid control education. In *IFAC Conference on Advances in PID Control*, volume 2, pages 691–696. Brescia, Italy, 2012. 204

J. Oravec and M. Bakošová. PIDDESIGN - Software for PID Control Education. In *IFAC Conference on Advances in PID Control*, 2012. URL https://bitbucket.org/oravec/piddesign/wiki/. 61

J. Oravec, S. Blažek, and M. Kvasnica. Simplification of explicit mpc solutions via inner and outer approximations. In M. Fikar and M. Kvasnica, editors, *Proceedings of the 19th International Conference on Process Control*, pages 389–394, Štrbské Pleso, Slovakia, June 18-21, 2013 2013. Slovak University of Technology in Bratislava, Slovak University of Technology in Bratislava. 112

J. Oravec, M. Klaučo, M. Kvasnica, and J. Löfberg. Optimal vehicle routing with interception of targets? neighbourhoods. In *European Control Conference 2015*, pages 2538–2543, Linz, Austria, 2015. URL http://www.kirp.chtf.stuba.sk/publication_info.php?id_pub=1632. 52

P. Van Overschee and B. De Moor. N4sid: Subspace algorithms for the identification of combined deterministic-stochastic systems. *Automatica*, 30(1):75–93, January 1994. ISSN 0005-1098. doi: 10.1016/0005-1098(94)90230-5. URL http://dx.doi.org/10.1016/0005-1098(94)90230-5. 61

S. Olaru P. Ahmadi-Moshkenani and T. A. Johansen. Further results on the exploration of combinatorial tree in multi-parametric quadratic programming. In *Proc. of the European Control Conference*, 2016. 87

G. Pannocchia and J.B. Rawlings. Disturbance models for offset-free model-predictive control. *AIChE journal*, 49(2):426–437, 2003. 218

S. Prajna and A. Jadbabaie. Safety verification of hybrid systems using barrier certificates. In *Hybrid Systems: Computation and Control*, pages 477–492. Springer, 2004. 166

S. Prajna, A. Jadbabaie, and G. Pappas. A framework for worst-case and stochastic safety verification using barrier certificates. *Automatic Control, IEEE Transactions on*, 52(8):1415–1428, 2007. 166

F. Qian, D.R. Dixon, G. Newcombe, L. Ho, J. Dreyfus, and P.J. Scales. The effect of ph on the release of metabolites by cyanobacteria in conventional water treatment processes. *Harmful Algae*, 39(0):253 – 258, 2014. ISSN 1568-9883. doi: http://dx.doi.org/10.1016/j.hal.2014.08.006. URL http://www.sciencedirect.com/science/article/pii/S1568988314001589. 197

J.J. Qin, M.H. Oo, K.A. Kekre, F. Knops, and P. Miller. Impact of coagulation ph on enhanced removal of natural organic matter in treatment of reservoir water. *Separation and Purification Technology*, 49(3):295 – 298, 2006. ISSN 1383-5866. doi: http://dx.doi.org/10.1016/j.seppur.2005.09.016. URL http://www.sciencedirect.com/science/article/pii/S1383586605003187. 197

S. J. Qin and T. A. Badgewell. A survey of industrial model predictive control technology. 11:733–764, 2003. 54, 222

S.J. Qin and T. A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733 – 764, 2003. ISSN 0967-0661. doi: http://dx.doi.org/10.1016/S0967-0661(02)00186-7. URL http://www.sciencedirect.com/science/article/pii/S0967066102001867. 54

D. Quevedo, J. De Dona, and G. Goodwin. Receding horizon linear quadratic control with finite input constraint sets. In *IFAC 15th Triennial World Congress*, 2002. 143

J. B. Rawlings and K. R. Muske. The stability of constrained receding horizon control. 38(10):1512–1516, October 1993. 71

Renewable Energy Policy Network for the 21st Century REN21. Renewables 2015 global status report, 2016. URL http://www.ren21.net/wp-content/uploads/2015/07/REN12-GSR2015_Onlinebook_low1.pdf. 213

J. Richalet, A. Rault, J.L. Testud, and J. Papon. Algorithmic control of industrial processes. *In Proceedings of the 4th IFAC symposium on identification and system parameter estimation.*, pages 1119–1167, 1976. 53

J. Richalet, A. Rault, J.L. Testud, and J. Papon. Model predictive heuristic control: Applications to industrial processes. *Automatica*, 14(5):413 – 428, 1978. ISSN 0005-1098. doi: http://dx.doi.org/10.1016/0005-1098(78)90001-8. URL http://www.sciencedirect.com/science/article/pii/0005109878900018. 53

P. O. M. Scokaert and J. B. Rawlings. Constrained Linear Quadratic Regulation. 43(8):1163–1169, August 1998. 71

S.Di Cairano, D. Yanakiev, A. Bemporad, I. Kolmanovsky, and D. Hrovat. Model predictive idle speed control: Design, analysis, and experimental evaluation. *IEEE Trans. Control Systems Technology*, 20(1):84–97, 2012. 74

L.R.E. Shead, K.R. Muske, and J.A. Rossiter. Conditions for which linear {MPC} converges to the correct target. *Journal of Process Control*, 20(10): 1243 – 1251, 2010. ISSN 0959-1524. doi: http://dx.doi.org/10.1016/j.jprocont.2010.09.001. URL http://www.sciencedirect.com/science/article/pii/S0959152410001812. 218

I. Silva, K. Richeson, B. Krogh, and A. Chutinan. Modeling and verifying hybrid dynamic systems using CheckMate. In *Proceedings of 4th International Conference on Automation of Mixed Processes*, pages 323–328, 2000. 167

J. Snoeyink. Handbook of discrete and computational geometry. chapter Point Location, pages 559–574. CRC Press, Inc., Boca Raton, FL, USA, 1997. ISBN 0-8493-8524-5. 100

J. Spjøtvold, P. Tøndel, and T. A. Johansen. A Method for Obtaining Continuous Solutions to Multiparametric Linear Programs. Prague, Czech Republic, 2005. 74

G. Stewart and F. Borrelli. A Model Predictive Control Framework for Industrial Turbodiesel Engine Control. pages 5704–5711, Cancun, Mexico, Dec 2008. 74

K. J. Åström and T. Hägglund. *PID Controllers - Theory, Design, and Tuning (2nd Edition)*. Instrument Society of America, Research Triangle Park, North Carolina, 1995. ISBN 978-1-55617-516-9. 203

O. Stursberg and B.H. Krogh. Efficient representation and computation of reachable sets for hybrid systems. In *Hybrid Systems: Computation and Control*, pages 482–497. Springer, 2003. 167

A. Sz

Hucs, M. Kvasnica, and M. Fikar. *Selected Topics in Modelling and Control*, chapter Data Compression Techniques for Complexity Reduction in Explicit

MPC, pages 18–23. Number 7. Slovak University of Technology Press Bratislava, 2011. 112

Inc. The MathWorks. R2016a documentation - control system toolbox - pidtool, 2016a. URL http://www.mathworks.com/help/control/ref/pidtool.html. 204

Inc. The MathWorks. R2016a documentation - control system toolbox - pidtune, 2016b. URL http://www.mathworks.com/help/control/ref/pidtune.html. 204

P. Tøndel and T.A. Johansen. Complexity reduction in explicit linear model predictive control. In *Proc. of 15-th IFAC world congress*, 2002. 222

P. Tøndel, T. A. Johansen, and A. Bemporad. Evaluation of piecewise affine control via binary search tree, 2002. 104

F. D. Torrisi. *Modeling and Reach-Set Computation for Analysis and Optimal Control of Discrete Hybrid Automata*. Dr. sc. thesis, Zürich, Switzerland, March 2003. 167

G. Valencia-Palomo and J.A. Rossiter. Using Laguerre functions to improve efficiency of multi-parametric predictive control. In *Proceedings of the American Control Conference*, pages 4731–4736, Baltimore, USA, 2010. 112

A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006. 54

R. Webster. *Convexity,*. Oxford, England: Oxford University Press,, 1995. 38

Ch. Wen, X. Ma, and B. E. Ydstie. Analytical expression of explicit MPC solution via lattice piecewise-affine function. *Automatica*, 45(4):910 – 917, 2009. ISSN 0005-1098. 112

P. Wieland and F. Allgöwer. Constructive safety using control barrier functions. In *Proceedings of the 7th IFAC Symposium on Nonlinear Control Systems*, pages 462–467, 2007. 166

H.P. Williams. *Model Building in Mathematical Programming*. John Wiley & Sons, Third Edition, 1993. 175

L.B. Willner. On Parametric Linear Programming. *SIAM Journal on Applied Mathematics*, 15(5):1253–1257, September 1967. 74, 167

L.A. Wolsey. *Integer Programming*. Wiley Series in Discrete Mathematics and Optimization. Wiley, 1998. ISBN 9780471283669. 49

G. M. Ziegler. *Lectures on Polytopes*. Springer, 1994. 123

G.M. Ziegler. *Lectures on polytopes*. Graduate texts in mathematics. Springer, New York, 1995. ISBN 0-387-94329-3. 38

Author's Publications

- Articles in journals indexed in Current Contents Database

    1. Holaza, J. – Takács, B. – Kvasnica, M. – Di Cairano, S.: Nearly optimal simple explicit MPC controllers with stability and feasibility guarantees. In *Optimal Control Applications and Methods*, vol.6, 2015.

- Article in IFAC World Congress proceedings

    1. Holaza, J. – Takács, B. – Kvasnica, M. : Verification of Performance Bounds for A-Posteriori Quantized Explicit MPC Feedback Laws. In *Preprints of the 19th IFAC World Congress Cape Town*, Cape Town, South Africa, pp. 1035–1040, 2014.

- Chapter or pages in book

    1. Holaza, J. – Takács, B. – Kvasnica, M. : Complexity Reduction in Explicit MPC via Bounded PWA Approximations of the Cost Function. In *Selected Topics in Modelling and Control, Editor(i): Mikleš, J., Veselý, V., Slovak University of Technology Press, vol. 8*, pp. 27–32, 2012.

- Articles in international conference proceedings

    1. Holaza, J. – Takács, B. – Kvasnica, M. : Safety Verification of Implicitly Defined MPC Feedback Laws. In *European Control Conference*, Linz, Austria, pp. 2552–2557, 2015.

2. Kvasnica, M. – Holaza, J. – Takács, B. – Ingole, D. : Design and Verification of Low-Complexity Explicit MPC Controllers in MPT3. In *European Control Conference*, Linz, Austria, pp. 2600–2605, 2015.

3. Kvasnica, M. – Takács, B. – Holaza, J. – Di Cairano, S. : On Region-Free Explicit Model Predictive Control. In *54rd IEEE Conference on Decision and Control*, Osaka, Japan, pp. 3669–3674, 2015.

4. Holaza, J. – Takács, B. – Kvasnica, M. : Simple Explicit MPC Controllers Based on Approximation of the Feedback Law. In *ACROSS Workshop on Cooperative Systems*, Zagreb, Croatia, pp. 48–49, 2014.

5. Takács, B. – Holaza, J. – Kvasnica, M. – Di Cairano, S. : Nearly-Optimal Simple Explicit MPC Regulators with Recursive Feasibility Guarantees. In *IEEE Conference on Decision and Control*, Florence, Italy, pp. 7089–7094, 2013.

6. Takács, B. – Holaza, J. – Kvasnica, M. : NLP-based Derivation of Bounded Convex PWA Functions: Application to Complexity Reduction in Explicit MPC. In *Veszprém Optimization Conference: Advanced Algorithms*, Veszprém, Hungary, pp. 95–95, 2012.

- Articles in domestic conference proceedings

1. Kalúz, M. – Holaza, J. – Janeček, F. – Blažek, S. – Kvasnica, M. : A Robotic Traffic Simulator for Teaching of Advanced Control Methods. In *Proceedings of the 11th IFAC Symposium on Advances in Control Education*, Bratislava, Slovakia, 2016 (accepted).

2. Sharma, A. – Drgoňa, J. – Ingole, D. – Holaza, J. – Valo, R. – Koniar, S. – Kvasnica, M. : Teaching Classical and Advanced Control of Binary Distillation Column. In *Proceedings of the 11th IFAC Symposium on Advances in Control Education*, Bratislava, Slovakia, 2016 (accepted).

3. Ingole, D. – Holaza, J. – Takács, B. – Kvasnica, M. : FPGA-Based Explicit Model Predictive Control for Closed-Loop Control of Intravenous Anesthesia. In *Proceedings of the 20th International Conference on Process Control*, Štrbské Pleso, Slovakia, pp. 42–47, 2015.

4. Kvasnica, M. – Takács, B. – Holaza, J. – Ingole, D. : Reachability Analysis and Control Synthesis for Uncertain Linear Systems in MPT.

In *Proceedings of the 8th IFAC Symposium on Robust Control Design*, Bratislava, Slovakia, pp. 302–307, 2015.

5. Takács, B. – Holaza, J. – Števek, J. – Kvasnica, M. : Export of Explicit Model Predictive Control to Python. In *Proceedings of the 20th International Conference on Process Control*, Štrbské Pleso, Slovakia, pp. 78–83, 2015.

6. Holaza, J. – Takács, B. – Kvasnica, M. : Synthesis of Simple Explicit MPC Optimizers by Function Approximation. In *Proceedings of the 19th International Conference on Process Control*, Štrbské Pleso, Slovakia, pp. 377–382, 2013.

Curriculum Vitae

Juraj Holaza

Date of Birth: January 31, 1988
Citizenship: Slovakia
Email: juraj.holaza@stuba.sk
Homepage: http://www.kirp.chtf.stuba.sk/ holaza/

Education

- B.S. Process Control, Slovak University of Technology, 2010.
  - *Minors*: Optimal Process Control and Chemical Engineering.
- M.S. Process Control, Slovak University of Technology, 2012
  - *Minors*: Model Predictive Control, Convex Optimization
- Ph.D. Process Control, Slovak University of Technology, *expected* 2016
  - *Majors*: Fast and Memmory-Efficient Implementation of Model Predictive Control
  - *Minors*: Convex Optimization
  - *Advanced Study*: Implementation of model predictive control to wind turbine at University of Zagreb, Croatia, March 2016 – May 2016

Research Fields

- Model Predictive Control, Parametric Programming, Convex Optimization

Miscellaneous

Computer Skills

- C/C++, Matlab, Simulink, PHP, HTML, XML, Python
- Windows, Unix/Linux, MS Office, Keynote, LaTeX
- Siemens Simatic, Foxboro, WinCC

Language Skills

- English

Resumé

Predkladaná dizertačná práca sa snaží preskúmať možnosti rýchlej a pamäťovo nenáročnej implementácii prediktívneho riadenia za účelom rozšírenia tejto stratégie aj do nízkoúrovňových priemyselných riadiacich platforiem. Prediktívne riadenie predstavuje pokročilý prístup k riadeniu, ktorého popularita neustále rastie a svoje uplatnenie si nachádza v mnohých (najmä priemyselných) odvetviach. Medzi najväčšie výhody danej metódy patrí predikcia vývoja riadeného systému, na základe zostrojeného matematického modelu, a teda možnosť optimalizovať sekvenciu akčných zásahov na celom predikčnom horizonte, čo výraznou mierou zlepšuje celkovú kvalitu riadenia. Ďalej schopnosť navrhovať spätnoväzbové regulátory pre mnoho rozmerové systémy, pri súčasnom zohľadnení všetkých systémových a kvalitatívnych obmedzení, ktoré sú explicitne zakomponované vo výslednom optimalizačnom probléme. Tu by sme mohli zdôrazniť, že každý priemyselný proces ma svoje fyzikálne limitácie, ktoré vychádzajú priamo z jeho konštrukcie. A čo viac, v súčasnej dobe sa kladie veľký dôraz na splnenie výrobných noriem, ktoré predstavujú sériu prísnych požiadaviek ako už na kvalitu produktu, tak aj na celkovú bezpečnosť pri výrobnom procese. V konečnom dôsledku, sú to práve zmienené prednosti prediktívneho riadenia, ktoré umožňujú nielen znížiť materiálovú a energetickú spotrebu, ale taktiež minimalizovať negatívne dopady na životné prostredie a zabezpečiť bezpečnú prevádzku, pri súčasnej maximalizácii zisku.

Na druhej strane, ako sme aj uviedli v tabuľke 3.1, najväčšou implementačnou prekážkou prediktívneho riadenia je vysoká výpočtová náročnosť, takto zostrojených optimalizačných problémov, ktorá môže prevyšovať prípustné prostriedky cieleného riadiaceho hardvéru. Pre bližšie spresnenie, za účelom poskytnutia garancie

stability, optimality a dodržania všetkých ohraničení, celý optimalizačný problém musí byť vypočítaný do doby jednej periódy vzorkovania. Avšak dodržanie tejto požiadavky môže byť ohrozené, ak cielený riadiaci hardvér má obmedzené výpočtové prostriedky, alebo ak dynamika riadeného procesu si vyžaduje príliš rýchlu frekvenciu vzorkovania. A preto, aby sme predišli tejto situácii, dodatočné opatrenia musia byť vykonané, ktoré znížia požiadavky prediktívneho riadenia na takú mieru, aby bola umožnená úspešná implementácia. Predkladaná práca sa zaoberá touto problematikou a poukazuje na možné riešenia ako daný cieľ docieliť.

Jednou z možností, ako znížiť výpočtové nároky prediktívneho riadenia je využitie konceptu explicitného prediktívneho riadenia. Hlavnou myšlienkou uvedeného prístupu je využitie parametrického programovania za cieľom dopredného vypočítania celého optimalizačného problému a to pre všetky prípustné začiatočné podmienky riadeného procesu. Výsledkom parametrického programovania je po častiach afínna (PWA) funkcia, ktorá mapuje stavy systému na optimálne akčné zásahy. Ilustračný príklad PWA zákonu riadenia je zobrazený na obrázku 4.1(b) a jeho polytopická partícia (doména funkcie) je znázornená na obrázku 4.1(a). Následnou implementáciou explicitného riešenia do riadiaceho hardvéru, sú výpočtové nároky znížené na jednoduché matematické operácie, ktoré môžu byť efektívne vykonané aj na nízkoúrovňových riadiacich platformách. Avšak cena, ktorú musíme zaplatiť za takéto rýchle vyhodnocovanie akčných vstupov, je zvýšená pamäťová záťaž, ktorá je priamo úmerná počtu regiónov tvoriacich polytopickú partíciu. Tu by sme mali zdôrazniť, že zatiaľ čo zložitosť explicitných riešení (v zmysle počtu regiónov) rastie exponencionálne s dĺžkou predikčného horizontu, tak dostupný pamäťový potenciál priemyselných hardvérov (akými sú napríklad mikročip a programovateľný logický automat) je striktne limitovaný iba do niekoľkých kilobajtov. Výhody vyplývajúce z explicitného prediktívneho riadenia prilákalo mnohých výskumníkov, ktorí poukázali na viacero možnosti ako znížiť počet regiónov a teda ako zmierniť pamäťové požiadavky tejto metodológie. V literatúre je tento problém vedený ako redukcia zložitosti explicitných prediktívnych zákonov riadenia a prvým prínosom dizertačnej práce je obohatenie tejto oblasti o jednu novú techniku.

Navrhnutá metóda redukcie zložitosti explicitných prediktívnych zákonov riadenia predpokladá, že zložité explicitné riešenie $\mu(x)$ je dostupné. Základnou myšlienkou je nahradiť $\mu(x)$ za jej jednoduchšiu verziu $\widetilde{u}(\cdot)$ pričom: 1.) $\widetilde{u}(\cdot)$ bude garantovať rekurzívnu riešiteľnosť; 2.) $\widetilde{u}(\cdot)$ zabezpečuje asymptotickú stabilitu uzavretého

systému; 3.) chyba predstavujúca integrál štvorca odchyliek medzi funkciami $\mu(\cdot)$ a $\widetilde{u}(\cdot)$ (respektíve suboptimalita $\widetilde{u}(\cdot)$ vzhľadom na $\mu(\cdot)$) je minimalizovaná. Uvedený problém je ilustrovaný na obrázku 5.1. Na zostrojenie aproximovaného zákona riadenia $\widetilde{u}(\cdot)$ je potrebné najskôr zostrojiť polytopické regióny, nad ktorými bude $\widetilde{u}(\cdot)$ definovaný, a následne zostrojiť lokálne afínne členy pre každý jeden nový región. K prvej úlohe sme pristúpili opätovným vyriešením rovnakého optimalizačného problému (ktorým bol získaný $\mu(x)$), avšak s nižším predikčným horizontom. Takto sme získali nový explicitný regulátor $\widehat{\mu}(\cdot)$, ktorý bol definovaný nad jednoduchšou polytopickou partíciou. Avšak so znížením zložitosti sa výrazne zhoršila aj výkonnosť takto získaného regulátora $\widehat{\mu}(\cdot)$, v porovnaní s $\mu(x)$. Preto, za účelom zmiernenia straty na výkonnosti, sme si ponechali iba polytopickú partíciu $\widehat{\mu}(\cdot)$ a aproximovali sme nad ňou nové lokálne zákony riadenia funkcie $\widetilde{u}(\cdot)$, ktoré minimalizovali integrál sumy odchyliek $\widetilde{u}(\cdot)$ od $\mu(x)$. A čo viacej, pri aproximačnom probléme sme si vynútili, aby nové explicitné riešenie $\widetilde{u}(\cdot)$ spĺňalo všetky pôvodné ohraničenia a nútilo uzavretý systém asymptoticky klesať do počiatku. Účinnosť metódy bola demonštrovaná na viacerých príkladoch. Na základe výsledkov, kompaktne zdokumentovaných v tabuľkách 5.1, 5.2 a 5.3, môžeme prísť k záveru, že navrhnutá technika dokáže efektívne znížiť zložitosť explicitných spätnoväzbových zákonov riadenia, a to iba za cenu mierneho poklesu na ich výkonnosti.

Rýchla a jednoduchá (nie sú potrebné žiadne drahé matematické programy pre riešenie optimalizačných problémov) implementácia nie je jedinou výhodou explicitného prediktívneho riadenia. Parametrické riešenie nám ponúka, prostredníctvom PWA funkcie, priamy prehľad nad závislosť medzi nameranými stavmi a optimálnymi akčnými zásahmi, čo nám umožňuje podrobne analyzovať vlastnosti zostrojeného zákonu riadenia. Táto prednosť bola využitá pri druhom prínose predkladanej práce, kde sme preskúmavali vplyv kvantizátora, na vlastnosti riadenia.

Riadenie pod kvantizovanou spätnou väzbou je v súčasnosti veľmi dôležitá oblasť, nakoľko väčšina riadiacich algoritmov je implementovaná na digitálnych platformách, ktoré sú (zo svojej podstaty) ovplyvnené kvantizačnému pravidlu vzhľadom na ich konečnú presnosť a prevodmi medzi analógovými a digitálnymi signálmi. Zostrojenie prediktívneho riadenia pre takéto systémy nie je vôbec ľahké a vo všeobecnosti vedie ku komplikovanej matematickej formulácií optimalizačného problému, čo zabraňuje ich následnú implementáciu do štandardných priemyselných riadiacich platforiem. Preto sme v našom prístupe navrhli vynechať informáciu o kvantizovaných akčných zásahov z optimalizačného problému, čím sme

zostrojili oveľa jednoduchší prediktívny zákon riadenia. Následne sme poskytli rigorózne certifikáty, ktoré overili, či takýto jednoduchý regulátor spĺňa všetky dôležité riadiace vlastnosti, a to aj s ohľadom na zaokrúhľovacie kvantizačné pravidlo. Takýmto spôsobom sme schopní zistiť, či navrhnutý regulátor, ktorého výpočtové nároky sú znížené, je možné (bezpečné) implementovať do uzavretého regulačného obvodu. A teda prostredníctvom takejto spätnej verifikácie sme schopní zistiť, či navrhnutý regulátor, ktorého výpočtové nároky sú znížené, je možné (bezpečné). Pri zostrojovaní certifikátu sme postupovali nasledovne: 1.) Odvodili sme si explicitné riešenie prediktívneho riadenia $\mu(x)$. 2.) Prostredníctvom známych kvantizačných hodnôt q_i a s použitím techniky voronojovho diagramu sme k pôvodnému zákonu riadenia $\mu(x)$ našli jeho príslušnú kvantizovanú verziu $\mu(\cdot)$. 3.) S využitím parametrického programovania sme zostrojili hornú $\overline{V}$ a spodnú $\underline{V}$ hranicu, ktoré vymedzovali priestor stavov a akčných zásahov, kde určitá riadiaca vlastnosť je splnená. Konkrétne, v našej práci sme ukázali, ako zostrojiť $\overline{V}$ a $\underline{V}$ pre overenie rekurzívnej riešiteľnosti, asymptotickej stability uzavretej slučky a dodržanie degradácie výkonnosti regulátora. 4.) Predložili sme lineárny optimalizačný problém, ktorý zisťuje, či $\mu(\cdot)$ je ohraničený $\overline{V}$ a $\underline{V}$. Ilustračný obrázok takejto certifikácie je znázornený na obrázku 6.1. Aplikovateľnosť takéhoto prístupu bolo preukázané prostredníctvom troch názorných príkladov, pričom výsledky jednotlivých verifikácii sú zdokumentované v tabuľkách 6.1, 6.2 alebo 6.3.

Certifikácia bezpečnosti prediktívnych zákonov riadenia bolo objektom skúmania aj v tretom prínose predkladanej dizertačnej práci. Konkrétne sme sa zamerali overiť, či prediktívny regulátor je navrhnutý tak, aby prinútil riadený systém (inicializovaný z vopred známych počiatočných podmienok) sa vyhnúť všetkým nebezpečným stavom. Takýto problém dosiahnuteľnosti je znázornený na obrázku 6.9. Je nutné ale pripomenúť, že prediktívne riadenie dovoľuje zahrnúť všetky obmedzenia požiadavky priamo do optimalizačného problému. Avšak niektoré bezpečnostné špecifikácie (akými sú napríklad vyhýbanie sa prekážkam, či limitácie na preregulovanie a čas ustálenia riadenej veličiny) je veľmi ťažké zakomponovať, nakoľko vedú k zložitým (nekonvexným) matematickým, ktoré si vyžadujú zvýšené výpočtové nároky. Preto, za účelom zníženia implementačných požiadaviek, navrhujeme tieto reštrikcie vynechať a spätne overiť ich platnosť prostredníctvom certifikátov. Na rozdiel od predchádzajúcej metódy poukazujeme, že takúto verifikáciu je možné vykonať aj bez analytického riešenia optimalizačného problému (resp. bez nutnosti použitia parametrického programovania). Naopak, Karush-Kuhn-Tucker

(KKT) podmienky sú využite na charakterizáciu optimálnych akčných zásahov. Boli prezentované tri principiálne formulácie, každá z nich v podobe celočíselného lineárneho programu, ktoré poskytovali jednoznačnú odpoveď na otázku, či prediktívne riadenie preukazuje všetky potrebné bezpečnostné vlastnosti. Uviedli sme aj, že za predpokladu existencie pozitívnej invariantnej terminálnej množiny môžeme poskytnúť certifikát pre nekonečne veľa krokov. Systému štyroch zásobníkov kvapalín bol využitý na ilustráciu všetkých troch formulácií navrhnutej metódy. Výsledky jednotlivých certifikátom sú znázornené na obrázkoch 6.12 - 6.14. Poukázali sme aj na alternatívne využitie verifikačnej metódy, ktoré sme postavili do pozície techniky redukcie zložitosť explicitných prediktívnych zákonov riadenia, ktorej účinnosť je zdokumentovaná v tabuľke 6.4 a znázornená na obrázku 6.8.

Posledná časť predkladanej prace bola venovaná praktickej aplikácii prediktívneho riadenia. Boli riadene dva odlišné procesy.

V prvom z nich sme sa zamerali na riadenie pH v chemickom reaktore, kde prebiehala reakcia medzi kyselinou octovou a hydroxidom sodným. Našou úlohou bolo navrhnúť spätnoväzbový regulátor, ktorý by sa vysporiadal so silne nelineárnou závislosťou medzi pH a koncentráciami kyseliny a zásady v reaktore. Táto relácia je známe širokej komunite titračnou krivkou, ktorá nadobúda zakrivený tvar 'S'. Ďalšou požiadavkou bolo poskytnutie garancii na dodržovanie stanovených systémových ohraničení. Na dosiahnutie tohto cieľa sme najskôr daný systém identifikovali na základe vstupno-výstupných signálov. Následné sme zostrojili PID regulátor, ktorý prostredníctvom prietoku zásaditého roztoku ovplyvňoval pH v chemickom reaktore. Ďalej, za účelom zlepšenia riadiacej výkonnosti, sme pridali k pôvodnej spätnoväybovej slučke aj MPC regulátor. Jeho úlohou bolo poskytovať referencie PID regulátoru tak, aby všetky ohraničenia boli dodržané a preregulovania bolo potlačené. Tieto vlastnosti boli aj ilustrovane prostredníctvom priložených grafov.

Druhým procesom, ktorému sme sa venovali v tejto prací bola veterná turbína, ktorá bola riadená priamo zo strany generátora, kde našou úlohou bolo (aby sme dosiahli optimálnu produkciu energie) udržanie jeho operačnej rýchlosti na referenčnej hodnote. Našou úlohou bolo navrhnúť MPC regulátor pre FOC s napäťovo riadením prevodníkom, pričom riadiace napätie bolo podmienené bezpečnostnými ohraničeniam. Hlavnou výzvou tejto úlohy bolo vysporiadanie sa s kruhovými ohraničeniami a udržať rýchlu periódu vzorkovania rovnú 0.4 milisekúnd. Ukázali sme, že navrhnúť takýto zákon riadenia nebolo vôbec jednoduché. Konkrétne, na od-

stránenie trvalej regulačnej odchýlky sme zaviedli koncept modelovania odchýlky medzi predikčným modelom a reálnou dynamikou systému. Kruhové obmedzenia sme aproximovali sériou lineárnych ohraničení, resp. mnohostenom, a pre udržanie nízkej zložitosti optimalizačného problému sme zaviedli stratégiu move-blocking, pomocou ktorého sme znížili počet optimalizovaných premenných. Prostredníctvom grafov sme ilustrovali, že takto navrhnute MPC riadenie dosahuje lepšie riadiace výsledky ako konvenčný PID regulátor.

Pre ďalšie pokračovanie tejto práce sme uvažovali prepojiť navrhnutú metódu redukcie zložitosti explicitných prediktívnych regulátorov s ďalšími redukčnými technikami, za účelom dosiahnutia ešte väčšieho zníženia pamäťovej stopy takýchto zákonov riadenia. Pri prvej certifikačnej metóde sme sa zamerali na overovanie výkonností prediktívnych regulátorov, ktoré boli charakterizované prostredníctvom PWA hraníc. Túto techniku by sme chceli ďalej rozšíriť aj na iné typy funkcií (napríklad polynomiálne). Rovnako aj druhý verifikační prístup by sme chceli zovšeobecniť. Konkrétne by sme chceli zahrnúť do certifikácie aj vplyvy aditívnych porúch a parametrických neurčitostí, ktoré sú prítomne pri každom reálnom spätnovázbovom riadení. Ďalej by sme ešte chceli využiť potenciál tejto metódy aj z pohľadu redukcie zložitosti prediktívnych explicitných regulátorov. Ako sme si mohli všimnúť, navrhnutá technika nepotrebuje poznať dopredu analytické riešenie prediktívneho optimalizačného problému k overeniu dosiahnuteľnosti stanovenej nebezpečnej množiny. Táto vlastnosť by sa mohla využiť už pri algoritmoch generujúcich explicitné zákony riadenia, čo by nám priamo umožnilo zostrojovať jednoduchšie explicitné regulátory. Pri riadení pH procesu by sme chceli zahrnúť aj neurčitosti systému a aditívne poruchy do návrhu prediktívneho riadenia. Rovnako by sme chceli znížiť zložitosť takto navrhnutého regulátora a implementovať ho do programovateľných automatov (PLC). A nakoniec, z pohľadu riadenia veternej turbíny, by sme chceli pridať dodatočné bezpečné ohraničenia, ktoré by (v prípade už poškodenej turbíny) zabraňovali ďalšiemu jej ničeniu. Týmto by sme dokázali životnosť veternej turbíny ešte ďalej predlžiť, čo by viedlo k efektívnejšej produkcii energie.