SLOVAK UNIVERSITY OF TECHNOLOGY
IN BRATISLAVA

FACULTY OF CHEMICAL AND FOOD TECHNOLOGY

Reg. No.: FCHPT-4622-82048

Structure-Exploiting Model Predictive Control for Real-Time and Embedded Applications

DISSERTATION THESIS

2025 **Ing. Kristína Fedorová**

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA

FACULTY OF CHEMICAL AND FOOD TECHNOLOGY

Reg. No.: FCHPT-4622-82048

Structure-Exploiting Model Predictive Control for Real-Time and Embedded Applications

DISSERTATION THESIS

Study programme:	Process Control
Study field:	Cybernetics
Training workspace:	Institute of Information Engineering, Automation, and Mathematics
Thesis supervisor:	prof. Ing. Michal Kvasnica, PhD.

2025 **Ing. Kristína Fedorová**

Slovak University of Technology in Bratislava
Department of Information Engineering and Process Control

Faculty of Chemical and Food Technology
Academic year: 2024/2025
Reg. No.: FCHPT-4622-82048

STU FCHPT

DISSERTATION THESIS TOPIC

Student: **Ing. Kristína Fedorová**
Student's ID: 82048
Study programme: Process Control
Study field: Cybernetics
Thesis supervisor: prof. Ing. Michal Kvasnica, PhD.
Head of department: prof. Ing. Miroslav Fikar, DrSc.
Workplace: Institute of Information Engineering, Automation, and Mathematics

Topic: **Structure-Exploiting Model Predictive Control for Real-Time and Embedded Applications**

Language of thesis: English

Deadline for submission of Dissertation thesis: 15. 06. 2025
Approval of assignment of Dissertation thesis: 09. 06. 2025
Assignment of Dissertation thesis approved by: prof. Ing. Miroslav Fikar, DrSc. – Chairperson of Field of Study Board

Acknowledgment

First and foremost, I would like to express my sincere gratitude to my supervisor, prof. Ing. Michal Kvasnica, PhD., for his continuous support, guidance, and inspiration throughout the course of my studies. His encouragement and the opportunities he provided have been invaluable.

I would also like to extend my heartfelt thanks to doc. Ing. Juraj Oravec, PhD., for his expertise, insightful discussions, and support in tackling scientific challenges, all of which greatly contributed to shaping this thesis.

My deepest appreciation goes to my colleague, Yuning Jiang. Collaborating with you has been one of the most enriching and rewarding experiences of my doctoral journey. Thank you for your ideas, shared knowledge, and for helping me grow both as a researcher and as a person.

Special thanks to my colleagues from the Institute of Information Engineering, Automation, and Mathematics for their help, cooperation, and support over the years. I am especially grateful to my colleagues from office NB633 for all the scientific and not-so-scientific discussions, the team-building moments, their honest support, and for always helping me stay grounded.

Rada by som poďakovala svojim rodičom, Kataríne a Vladimírovi, za ich neochvejnú pomoc a podporu počas celého môjho štúdia. Ďakujem.

I am deeply grateful to my partner, Markus, for standing by my side every step of the way—thank you for your patience, encouragement, and love, especially during the most challenging moments.

Last but certainly not least, I would like to thank my sister, Viktória, whose early encouragement and guidance paved the way for my academic journey and who has supported me ever since.

Once again, thank you all!

Kristína Fedorová, 2025

Abstract

This dissertation addresses the challenge of implementing Model Predictive Control (MPC) in real-time applications on embedded hardware and on large-scale systems. Although MPC offers a framework for handling multivariable systems with constraints, its real-time implementation is often challenging due to the computational demands of online optimization. To overcome these limitations, this work presents three complementary contributions that focus on exploiting structure in real-time MPC. The first contribution introduces a generalized stopping criterion for first-order optimization methods used in linear-quadratic MPC with input constraints. This criterion ensures that the resulting suboptimal control action still guarantees the asymptotic stability of the closed-loop system, even with a fixed number of iterations. The proposed method is applicable to a broad class of linearly convergent solvers and significantly reduces computational effort by over 70%, with suboptimality below 4%. Its practical effectiveness is demonstrated through the implementation of projected gradient descent and ADMM methods. The second contribution proposes a parallelizable, suboptimal MPC framework tailored for structured systems with polytopic constraints. This approach combines the concept of distributed optimization with explicit MPC. It integrates them into an online scheme that ensures the asymptotic stability of the resulting closed-loop system. A novel dualization of the initial condition enhances recursive feasibility guarantees and reduces memory requirements. The resulting open-source `C`-code toolbox, `ParExMPC`, demonstrates scalability to high-dimensional problems and superior runtime performance compared to state-of-the-art implicit MPC solvers, with implementation also on embedded hardware platforms. The third contribution presents a distributed MPC scheme that utilizes a relaxed, recentered logarithmic barrier function combined with a fixed-iteration solver. The strategy introduces a spatial and temporal decomposition into a globally convergent distributed MPC algorithm, while maintaining a fixed computational complexity. The framework supports decentralized execution and is validated on a coupled spring-damper system, achieving significant runtime reductions compared to conventional approaches. The results presented in this thesis provide scalable, implementable, and theory-based strategies for real-time MPC under computational and structural constraints.

Abstrakt

Táto dizertačná práca sa zaoberá návrhom a implementáciou prediktívneho riadenia (MPC z anglického *Model Predictive Control*) na riadenie komplexných dynamických systémov s ohraničeniami v reálnom čase. Cieľom práce je predstaviť algoritmy na riešenie MPC optimalizačného problému, ktoré sú implementovateľné v reálnom čase aj na výpočtovo obmedzených hardvérových platformách. Tieto algoritmy sú založené na distribuovanej optimalizácii, ktorú umožňuje separovateľná štruktúra MPC problému. V prvej časti práce je predstavené všeobecné stopovacie kritérium pre optimalizačné metódy prvého rádu, použité pri riešení lineárneho MPC s obmedzeniami na vstupné veličiny. Toto kritérium, formulované ako pevný počet iterácií, zaručuje asymptotickú stabilitu uzavretého regulačného obvodu aj v prípade suboptimálneho akčného zásahu. Výsledkom je riadiaca schéma s dopredu známou výpočtovou zložitosťou, ktorá preukázateľne redukuje výpočtovú náročnosť až o približne 70% pri zachovaní poklesu kvality riadenia do 4%. V druhej časti práce predstavujeme návrh paralelizovateľného suboptimálneho MPC algoritmu, ktorý využíva uvedené stopovacie kritérium. Tento prístup kombinuje explicitné riadenie, kde je riešenie MPC problému predpočítané offline, s distribuovanou optimalizáciou, čo vedie k zníženiu offline zložitosti a eliminácii závislosti na dimenziách pôvodného problému. Práca zároveň navrhuje rozšírenie tejto metódy, ktoré znižuje pamäťové nároky vďaka modifikovanému spracovaniu počiatočnej podmienky. Výsledkom je softvérový nástroj `ParExMPC`, implementovaný v jazyku `C`, ktorý dosahuje nadštandardné výpočtové časy v porovnaní so známymi implicitnými MPC riešeniami, pričom je škálovateľný na veľkorozmerné problémy a vhodný pre implementáciu na výpočtovo obmedzených zariadeniach. Posledná časť sa venuje návrhu distribuovaného MPC algoritmu, založeného na relaxovanej logaritmickej bariérovej funkcii v kombinácii s pevným počtom iterácií. Tento prístup využíva nielen časovú, ale aj priestorovú distribúciu MPC problému (z hľadiska stavového priestoru) a je implementovaný prostredníctvom globálne konvergujúcej optimalizačnej schémy s nízkymi nárokmi na komunikáciu medzi jednotlivými jednotkami. Výsledná metóda zaručuje stabilitu uzavretého regulačného obvodu pri pevnej výpočtovej zložitosti a umožňuje decentralizovanú implementáciu. Celkovo práca prezentuje škálovateľné, prakticky realizovateľné a teoreticky podložené prístupy k prediktívnemu riadeniu v reálnom čase.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Despite significant theoretical progress, real-time implementation of modern optimal control strategies on embedded platforms continues to face many challenges. This is especially true for large-scale systems, where computational and memory demands present substantial challenges. This dissertation introduces a set of novel concepts that either directly address these challenges or provide a foundation for developing efficient and scalable control strategies. All contributions are formulated within the well-established framework of Model Predictive Control (MPC).

MPC has become a cornerstone of modern control engineering due to its ability to handle multivariable systems and constraints through concepts of mathematical optimization as presented in Maciejowski (2000) and Mayne et al. (2000). Furthermore, the ability to predict future state evolution and flexibility makes it superior to more straightforward control strategies, such as the proportional-integral-derivative (PID) controllers (Salem and Mosaad, 2015), especially in applications involving large-scale systems or those requiring constraint satisfaction.

Even with substantial academic progress and increasing industrial relevance, the widespread deployment of MPC in embedded systems remains limited. Although computational capabilities have improved dramatically, many embedded applications still face strict computational and memory limitations that prevent the use of generic optimization-based controllers in real time, as explored by Xi et al. (2013) and Forbes et al. (2015). This practical bottleneck has triggered a shift in research focus from purely theoretical to application-driven control policy design, emphasizing computational tractability and real-time feasibility, as explained in Schwenzer et al. (2021). In particular, there is increased interest in developing MPC algorithms suitable for large-scale systems, where traditional solvers often struggle with computational or memory limitations.

In general, we extinguish two types of strategies to solve MPC problems: (i) implicit

and (ii) explicit methods. Implicit approaches solve the MPC optimization problem online at each control step. Among the most established implicit techniques are active-set methods (Bemporad, 2015), interior-point methods (Wright, 1997), and first- and second-order methods (Beck, 2017). Active-set methods are efficient for small to medium-sized problems due to their ability to exploit sparsity. However, they lack scalability and robustness for larger systems. Moreover, while active-set solvers are associated with conservative exponential worst-case runtime bounds, practical performance often significantly exceeds these theoretical estimates (Ferreau et al., 2014). In contrast, interior-point methods are well-suited for large-scale problems and offer robustness, although they can be challenging to warm-start in closed-loop applications. For convex quadratic programs (QPs), interior-point methods exhibit polynomial worst-case runtime (Boyd and Vandenberghe, 2004), but these bounds tend to be overly conservative and fail to reflect practical performance. First-order methods are attractive due to their low computational demand per iteration and favorable scaling properties. However, their slow convergence limits their use in real-time applications. In contrast, second-order methods demonstrate superior convergence rates but are associated with substantially higher computational costs per iteration. Notably, both first- and second-order methods are inherently robust to early termination, allowing the design of suboptimal real-time MPC schemes where iterations are deliberately reduced.

In this thesis, we exploit the robustness of early termination of first- and second-order optimization methods. We develop a fixed-iteration stopping criterion that not only enables real-time implementation of MPC with fixed computational and memory requirements but also guarantees asymptotic stability of the resulting closed-loop system. Although the use of first-order methods in real-time MPC has been widely studied in recent years, the contributions have primarily focused on specific algorithms. The first group of such approaches is based on stopping the optimization algorithm once a predefined suboptimality tolerance is achieved. For example, in Giselsson and Rantzer (2014), a suboptimality-based stopping criterion was proposed for the Alternating Direction Method of Multipliers (ADMM) within the MPC framework, with formal stability guarantees. Similarly, Köhler et al. (2019) investigated a tolerance-based approach for distributed dual gradient methods, and Giselsson and Rantzer (2010) considered a related stopping strategy for dual decomposition methods. An alternative research direction aims to derive a priori lower bounds on the number of iterations required to guarantee stability, leading to fixed-iteration real-time MPC schemes. For example, the stopping criteria for the dual accelerated projected gradient methods are introduced in Rubagotti et al. (2014) and in Liao-McPherson et al. (2022). The stopping strategy for projected gradient methods with state constraints is presented in Leung et al. (2021). Additionally, stopping criteria for fast alternating minimization

algorithms are discussed in Yang et al. (2023). This thesis presents the generalized stopping criterion proposed in our work Fedorová et al. (2023), which establishes a fixed-iteration scheme for any first-order optimization algorithm guaranteeing the asymptotic stability of the resulting closed-loop system.

An alternative to implicit MPC is the class of explicit methods, which aim to shift the computational burden from the online phase to the offline phase. In explicit MPC, introduced in Bemporad and Morari (2002), the control law is precomputed by solving a multi-parametric optimization problem offline. The resulting solution is stored in the form of piecewise affine (PWA) maps defined over polyhedral regions of the state space. During online execution, the control input is obtained by evaluating the precomputed control law corresponding to the current system state. This approach significantly reduces online computational effort, which depends solely on assessing the appropriate region. The main advantage of explicit MPC lies in its ability to provide optimal control inputs with certified performance. However, the approach is limited by scalability issues. In particular, the number of regions required to represent the solution may grow exponentially with the number of constraints or the prediction horizon, resulting in excessive memory requirements and offline computational effort. These challenges have motivated the development of numerous techniques for reducing complexity (Oberdieck et al., 2016). For instance, regionless evaluation schemes have been proposed in Kvasnica et al. (2015), while other approaches have aimed to preserve optimality under reduced complexity by refining the constraint handling (Kvasnica and Fikar, 2012; Kvasnica et al., 2013). Approximate explicit MPC strategies, which store suboptimal control laws instead of exact solutions, have also been investigated in Kvasnica et al. (2011). Despite successful implementations in embedded systems (Pappas et al., 2021), explicit MPC remains tractable primarily for problems of moderate size. For larger-scale systems, it is often outperformed by efficient online solvers, such as those presented in Ferreau et al. (2014) and Frison et al. (2014), which offer better scalability and flexibility.

One of the recent directions in real-time MPC that is adopted in this thesis is to reduce computational burden by breaking down the original optimization problem into smaller, easier-to-solve subproblems. This idea has driven much of the current research on distributed MPC (Conte et al., 2016), which has proven effective not only in centralized settings but also in controlling network-structured systems. Most distributed MPC algorithms are based on either first-order or second-order optimization techniques (Farokhi et al., 2014). First-order methods, such as ADMM) (Boyd et al., 2011), rely on gradient information and are relatively simple to implement. In contrast, second-order methods, including Riccati-based approaches (O'Donoghue et al., 2013; Nielsen and Axehill, 2019) and Newton-type algorithms (Kozma et al., 2013; Ferreau

et al., 2014), typically offer a faster convergence rate by exploiting Hessian information. A clear advantage of these methods is that the subproblems can be solved in parallel, which is especially beneficial for large-scale systems. However, as mentioned before, slow convergence often limits the practicality of these iterative methods.

This thesis builds on the framework introduced in Jiang et al. (2021), which combines the strengths of distributed and explicit MPC. It addresses one of the primary limitations of explicit MPC, namely the exponential growth in complexity associated with the prediction horizon. The core idea is to decompose the original MPC problem into smaller subproblems that can be solved offline and efficiently evaluated online. As a consequence, this strategy significantly reduces offline complexity and extends the practical applicability of explicit MPC to more complex systems. We extend the method from Jiang et al. (2021) in several important directions. First, we guarantee the recursive feasibility of the resulting control scheme. Second, we reinforce the theoretical foundation for asymptotic stability by proposing a fixed-iteration real-time strategy, which relies on the generalized stopping criterion introduced in Fedorová et al. (2023). This criterion is tailored to the Augmented Lagrangian-based Alternating Direction Inexact Newton (ALADIN) method (Houska et al., 2016), a second-order distributed optimization algorithm known for its favorable convergence properties (Houska and Jiang, 2021). Furthermore, we demonstrate that our proposed enhancements reduce both the offline computational burden and, more importantly, the memory requirements associated with storing precomputed solutions. Moreover, we introduce our open-source toolbox `ParExMPC`, developed based on this methodology. As shown in our recent work Jiang et al. (2025b), `ParExMPC` performs competitively with established general-purpose MPC solvers.

Finally, many of the proposed stopping criteria, including our own, offer theoretical guarantees of closed-loop stability only in the presence of input (control action) constraints. In general, stability under state constraints is often enforced by constraint tightening, a common technique that ensures feasibility and stability but tends to result in overly conservative control behavior (Giselsson and Rantzer, 2014; Köhler et al., 2019; Shi et al., 2022; Yang et al., 2023). In some cases, this conservatism can prevent the controller from finding the optimal solution or degrade overall performance. To address this limitation, we propose an alternative strategy that avoids constraint tightening and instead strengthens the theoretical guarantees of the generalized stopping criterion (Fedorová et al., 2023) in the presence of state constraints. Specifically, we integrate a relaxed recentered logarithmic barrier (RRLB) function introduced in Feller and Ebenbauer (2018) into the distributed MPC framework, which is solved using the ALADIN algorithm. Our results demonstrate that the resulting subproblems, now free of explicit inequality constraints, can be solved efficiently, even without

parallelization, while still ensuring the asymptotic stability of the closed-loop system. Furthermore, we propose a novel formulation of distributed MPC in which the problem is decomposed not only across the prediction horizon (temporal domain) but also over the state space (spatial domain). The main advantage of spatio-temporal distribution is its scalability: even for large-scale systems, the problem can be decomposed into low-complexity subproblems. This makes the approach well-suited for real-time control, where centralized methods are often too computationally demanding.

In summary, this thesis examines the design of real-time control policies for linear MPC, with a particular focus on large-scale and distributed systems. It builds upon recent advances in first- and second-order optimization, explicit and distributed control, to propose novel theoretical tools and practical algorithms that address the key bottlenecks of scalability in deployment on embedded hardware.

1.1 Thesis Objectives

This thesis focuses on the challenge of designing and implementing structure-exploiting linear MPC for real-time and embedded applications. The main objectives can be summarized as follows:

- **Formulation of a generalized stopping criterion for arbitrary first-order methods applied to linear MPC with input constraints.** The proposed criterion guarantees fixed and predictable computational and memory requirements for solving linear MPC problems with input constraints. Furthermore, it ensures asymptotic stability of the closed-loop system despite the use of suboptimal control actions.
- **Enhancement of a parallelizable linear MPC strategy.** A novel dualization of the initial condition is introduced to ensure recursive feasibility in parallel MPC. When integrated with the distributed explicit MPC framework, this modeling approach also leads to reduced memory requirements.
- **Proof of global linear convergence for the ALADIN algorithm.** A theoretical proof is provided for the global linear convergence of the ALADIN algorithm applied to distributed convex nonlinear problems with separable nonlinear objectives, linear constraints, and fixed scaling matrices.
- **Design of a novel MPC strategy based on spatio-temporal decomposition.** The proposed approach combines temporal decomposition over the

prediction horizon with spatial partitioning of the system dynamics, resulting in a distributed control scheme with reduced computational complexity and improved scalability.

- **Extension of the generalized stopping criterion to MPC problems with state constraints.** The stopping criterion is adapted to ensure asymptotic stability of the closed-loop system in the presence of state constraints.
- **Validation of the proposed strategies on benchmark systems.** All methods are thoroughly tested on established benchmark problems.
- **Development of the open-source `ParExMPC` toolbox.** A parallelizable software tool for solving MPC problems, built upon the proposed framework and tailored for efficient deployment on embedded platforms.

1.2 Thesis Structure

This thesis is organized as follows. Chapter 2 introduces the theoretical background of the thesis. Section 2.1 begins with an introduction to the MPC strategy, presenting the mathematical formulation, theoretical properties, and current challenges. Section 2.2 then presents the fundamentals of distributed optimization, followed by detailed discussions of the two algorithms employed in this work: ADMM in Section 2.2.1 and ALADIN in Section 2.2.2. In Section 2.3, we present the concept of distributed MPC, thereby completing the theoretical preliminaries.

The main contributions of the thesis are built upon the theoretical foundation of MPC and closely align with the author's published research. Chapter 3 presents the generalized stopping criterion approach proposed in Fedorová et al. (2023), including both theoretical analysis and numerical case studies. Chapter 4 introduces the novel open-source toolbox `ParExMPC`, which combines distributed and explicit MPC as proposed in Jiang, Fedorová, Su, Oravec, Houska, and Jones (2025b), with enhancements in theoretical guarantees and memory efficiency. Chapter 5 develops a new distributed real-time MPC framework for large-scale systems as presented in Jiang, Fedorová, Schwan, Oravec, and Jones (2025a). Each of Chapters 3–5 contains its preliminaries, theoretical derivation, case studies, and discussions.

Finally, the thesis concludes in Chapter 6. The list of the author's publications and Curriculum vitae are provided in Appendix A and Appendix B, respectively.

Chapter 2

Theoretical Background

This chapter establishes the theoretical foundations that support the developments and contributions presented in the subsequent chapters of this thesis. We begin by formalizing the general framework of discrete-time linear MPC for constrained systems in a receding horizon control fashion. A standard formulation of the MPC problem is presented, including the assumptions and structural properties commonly adopted in the field. Particular attention is paid to recursive feasibility and asymptotic stability, key properties that are revisited throughout this work. The second part of the chapter introduces the principles of distributed optimization, with a focus on two widely used algorithms: the Alternating Direction Method of Multipliers (ADMM) and the Augmented Lagrangian-based Alternating Direction Inexact Newton (ALADIN) method. These methods are known for their scalability and suitability for large-scale and embedded control applications. Finally, we present the concept of distributed and parallel MPC, highlighting its potential to address computational challenges in the real-time control of large-scale and interconnected systems. We summarize the current state of the art and outline the main challenges that motivate the research directions explored in this thesis.

2.1 Model Predictive Control

Model Predictive Control (MPC) is an advanced model-based control strategy in which the control input is computed by solving a finite-horizon optimal control problem at each control step. The formulation is based on utilizing a dynamic model to predict future system behavior over a specified prediction horizon. To enable accurate prediction, the current system state, in form of state measurement, is incorporated into the optimization problem as initial condition. The MPC problem also integrates explicit constraints on both input and state variables, allowing for the systematic constraint handling (Maciejowski, 2000; Rawlings et al., 2017).

The control objective is typically to minimize a cumulative cost function, often quadratic, that penalizes deviations from a desired reference trajectory as well as the activation of control inputs. Although solving MPC problem yields an entire sequence of predicted optimal control inputs, only the first element of this sequence is applied to the system (Mayne et al., 2000). At the next control step, a new state measurement is obtained, the prediction horizon is shifted forward, and the procedure is repeated. This closed-loop policy is known as the receding horizon control.

The key strength of MPC lies in its ability to manage multivariable systems with hard constraints while providing formal guarantees on feasibility and stability. These properties have led to its widespread adoption across diverse application areas, including chemical process control (Arumugasamy and Ahmad, 2012), energy systems (Tarragona et al., 2021), automotive and transportation (Di Cairano and Kolmanovsky, 2019), and robotics (Katayama et al., 2023).

However, the practical implementation of MPC in real-time settings, particularly on embedded platforms and for large-scale systems, remains limited by the computational cost of solving constrained optimization problem online. This highlights the need for efficient numerical methods and structure-exploiting algorithms that enable reliable real-time implementation without compromising theoretical guarantees, as discussed in Schwenzer et al. (2021).

2.1.1 Mathematical Formulation

A model of the system dynamics is a fundamental component of any MPC formulation. It enables the controller to forecast the system's future evolution over the prediction horizon and, based on this prediction, to compute control actions that optimize the control objective while satisfying pre-defined constraints. Here, we formally define the discrete-time model used to represent the dynamics of the controlled system. In the discrete-time setting, the system evolution is typically described by the following difference equation

$$x(t + T_\mathrm{s}) = f_\mathrm{sys}(x(t), u(t)), \tag{2.1}$$

where $x(t) \in \mathbb{R}^{n_\mathrm{x}}$ denotes the state vector, $u(t) \in \mathbb{R}^{n_\mathrm{u}}$ is the control input vector at time step t, and the function $f_\mathrm{sys} : \mathbb{R}^{n_\mathrm{x}} \times \mathbb{R}^{n_\mathrm{u}} \to \mathbb{R}^{n_\mathrm{x}}$ defines the (potentially nonlinear) system dynamics. The parameter $T_\mathrm{s} \in \mathbb{R}_+$ denotes the fixed sampling time between successive control updates.

In many practical applications, particularly when computational tractability is a concern, the system dynamics is modeled using a linear time-invariant (LTI) approximation.

Here, we define the discrete-time LTI system in the standard state-space form as

$$x(t + T_\mathrm{s}) = Ax(t) + Bu(t), \tag{2.2}$$

where $A \in \mathbb{R}^{n_\mathrm{x} \times n_\mathrm{x}}$ is the state transition matrix and $B \in \mathbb{R}^{n_\mathrm{x} \times n_\mathrm{u}}$ is the input matrix. This linear formulation is commonly used in MPC due to its simplicity and the existence of efficient solution methods for the associated optimization problem.

Generally, controlled systems have pre-defined operational bounds representing actuator limits, safety standards, and operational regulations. To ensure that the computed control actions remain feasible, it is essential to incorporate these constraints explicitly into the control design. Such considerations are naturally embedded in MPC framework by formulating the control problem with constraints on both state and input variables. These constraints are enforced at each step of the prediction horizon, thereby ensuring that generated trajectories satisfy the physical and operational limitations of the system.

To formalize constraint formulation within the MPC framework, we define the admissible sets for state and input trajectories as

$$x \in \mathbb{X}, \quad u \in \mathbb{U}, \tag{2.3}$$

where $\mathbb{X} \subseteq \mathbb{R}^{n_\mathrm{x}}$ and $\mathbb{U} \subseteq \mathbb{R}^{n_\mathrm{u}}$ denote the sets of feasible states and control inputs, respectively.

For the purposes of this thesis, we assume that the state and input constraints are defined by polyhedral sets of the form

$$\mathbb{X} = \{x \in \mathbb{R}^{n_\mathrm{x}} \mid G^\mathrm{x} x \leq d^\mathrm{x}\}, \quad \mathbb{U} = \{u \in \mathbb{R}^{n_\mathrm{u}} \mid G^\mathrm{u} u \leq d^\mathrm{u}\}, \tag{2.4}$$

where $G^\mathrm{x} \in \mathbb{R}^{m_\mathrm{x} \times n_\mathrm{x}}$, $G^\mathrm{u} \in \mathbb{R}^{m_\mathrm{u} \times n_\mathrm{u}}$, $d^\mathrm{x} \in \mathbb{R}^{m_\mathrm{x}}$, and $d^\mathrm{u} \in \mathbb{R}^{m_\mathrm{u}}$ define the inequality constraints in half-space representation. Here, m_x and m_u denote the number of linear inequalities imposed on the state and input vectors, respectively. This formulation is particularly convenient for integration into QP problems, which form the computational core of linear MPC algorithms. The use of polyhedral constraint sets ensures compatibility with efficient convex optimization solvers that provide systematic constraint satisfaction over the prediction horizon.

Finally, with system dynamics and constraints specified, the remaining component of the MPC problem is the control objective. The objective function defines the performance criteria to be optimized by the control input. In the discrete-time setting, it is typically formulated as a sum of stage costs accumulated over the finite prediction

horizon. In this work, we focus on the regulatory MPC problem, where the objective is to bring the system states to the origin while minimizing control effort.

Given a prediction horizon of length N, the objective function is defined as

$$\min_{x,u} \; l_N(x_N) + \sum_{k=0}^{N-1} l_k(x_k, u_k), \tag{2.5}$$

where $l_k(x_k, u_k)$ denotes the stage cost evaluated at each time step k, and $l_N(x_N)$ represents the terminal cost associated with the final predicted state x_N. The stage cost typically penalizes deviations from the desired operating point as well as the required control effort. The terminal cost is introduced to ensure long-term performance and closed-loop stability under appropriate assumptions.

In this work, we adopt a quadratic cost structure, which is standard in linear MPC due to its convexity and compatibility with commonly used QP solvers. The objective function is therefore given by

$$\min_{x,u} \; x_N^\top P x_N + \sum_{k=0}^{N-1} \left(x_k^\top Q x_k + u_k^\top R u_k \right), \tag{2.6}$$

where $Q \succeq 0$, $R \succ 0$, and $P \succeq 0$ are symmetric positive semidefinite or positive definite weighting matrices of appropriate dimensions. The matrices Q and R define the trade-off between state regulation and control effort at each step of the prediction horizon. The terminal weighting matrix P is essential for stability guarantees of the closed-loop system and will be discussed in detail in Subsection 2.1.2.

With the system dynamics, constraints, and objective function defined, the MPC problem can be formalized as

$$\begin{aligned}
V(x(t)) := \min_{x_k, u_k} \quad & x_N^\top P x_N + \sum_{k=0}^{N-1} \left(x_k^\top Q x_k + u_k^\top R u_k \right) && && \text{(2.7a)} \\
\text{s.t.} \quad & x_{k+1} = A x_k + B u_k, \; | \; \lambda_k, && k \in \{0, \dots, N-1\} && \text{(2.7b)} \\
& x_k \in \mathbb{X}, && k \in \{0, \dots, N-1\} && \text{(2.7c)} \\
& u_k \in \mathbb{U}, && k \in \{0, \dots, N-1\} && \text{(2.7d)} \\
& x_N \in \mathbb{X}_N, && && \text{(2.7e)} \\
& x_0 = x(t). && && \text{(2.7f)}
\end{aligned}$$

Here, the state and input constraints are enforced at each time step k of the finite control horizon using the constraint sets $\mathbb{X}$ and $\mathbb{U}$, as previously defined, and $\mathbb{X}_N$

denotes control-invariant terminal set. The optimal value function $V : \mathbb{F} \to \mathbb{R}_{\geq 0}$ maps each feasible initial state $x(t) \in \mathbb{X}$ to the corresponding minimum cost, where the feasible set is defined as $\mathbb{F} = \{x \in \mathbb{R}^{n_x} \mid V(x) < \infty\}$. Throughout this thesis, Lagrange multipliers associated with the system dynamics are denoted by λ_k for $k \in \{0, \ldots, N-1\}$. These dual variables are written immediately following the equality constraints in the optimization problem formulation and are essential for sensitivity analysis and the development of dual-based optimization methods.

The initial condition $x_0 = x(t)$ in problem (2.7) is given by the current measured or estimated state of the system. Solving the optimization problem yields the optimal control sequence $u^\star = [u_0^{\star\top}, u_1^{\star\top}, \ldots, u_{N-1}^{\star\top}]^\top$. In terms of receding horizon control strategy, only the first control input, $u(t) = u_0^\star$, is applied to the system. At the next sampling instant, the state is updated using new measurements or state estimates, and the optimization problem is solved again. This feedback mechanism enables the controller to adapt to disturbances and model mismatches in real-time.

2.1.2 Recursive Feasibility and Asymptotic Stability

To ensure both the feasibility of the optimization problem (2.7) and desirable stability properties of the resulting closed-loop system, specific standard properties of the MPC formulation must be observed.

A common approach is to select the terminal cost matrix P as the solution to the discrete-time algebraic Riccati equation

$$P = A^\top PA - (A^\top PB)(R + B^\top PB)^{-1}(B^\top PA) + Q, \tag{2.8}$$

which corresponds to the solution of infinite-horizon Linear Quadratic Regulator (LQR) and contributes to the desired stability properties of the closed-loop system.

However, the selection of P alone does not generally guarantee recursive feasibility or asymptotic stability. To achieve stability guarantees of closed-loop system, it is standard practice to impose a terminal state constraint of the form $x_N \in \mathbb{X}_N$, where $\mathbb{X}_N \subseteq \mathbb{X}$ denotes a positively invariant terminal set. This constraint ensures that the terminal state remains within a region where a known stabilizing control law exists, thereby providing both recursive feasibility and stability of the MPC strategy.

As an alternative to using a terminal constraint, stability can also be ensured by choosing a sufficiently large prediction horizon N. Under this approach, and assuming that the input and state constraints are strictly feasible (i.e., the admissible set has a non-empty interior), it can be shown that the finite-horizon cost $V(x(t))$ equals the

infinite-horizon cost $V_\infty(x(t))$, defined as (2.7) with $N = \infty$, for all initial conditions $x(t)$ within the domain of V_∞. The infinite-horizon value function V_∞ satisfies the stationary Bellman equation and serves as a Lyapunov function. Under these conditions, the resulting MPC controller ensures both recursive feasibility and asymptotic stability of the closed-loop system (Rawlings et al., 2017).

To formalize these properties, we adopt the following standard assumptions, commonly used in the literature (Mayne et al., 2000; Rawlings et al., 2009, 2017).

Assumption 1. *For the MPC problem defined in* (2.7)*, we assume that the following conditions hold:*

a) The pair (A, B) *is controllable.*

b) The constraint sets $\mathbb{X} \subset \mathbb{R}^{n_x}$*,* $\mathbb{U} \subset \mathbb{R}^{n_u}$*, and the terminal set* $\mathbb{X}_N \subset \mathbb{X}$ *are convex, closed, and contain the origin in their strict interiors.*

b) The terminal set $\mathbb{X}_N$ *is a positive control invariant set.*

d) The weighting matrices Q*,* R*, and* P *are symmetric and positive definite.*

e) The terminal cost $x_N^\top P x_N$ *serves as a Lyapunov function on the terminal set* $\mathbb{X}_N$*.*

Under Assumption 1, the receding horizon implementation of the MPC problem (2.7) guarantees both recursive feasibility and asymptotic stability of the closed-loop system. Specifically, recursive feasibility ensures that if the optimization problem is feasible at the initial time, it remains feasible at all subsequent time steps, provided the MPC control law is applied. Simultaneously, asymptotic stability guarantees that the closed-loop state trajectory converges to the origin as time progresses.

2.1.3 Current Challenges

Although MPC is theoretically well-established and widely applied in industry and safety-critical systems, its practical use remains challenging, especially when real-time execution is required. The main limitation is the need to solve a constrained, possibly large-scale, optimization problem in each control step. This becomes critical for systems with fast dynamics, long prediction horizons, or complex constraints, where the computational burden can compromise real-time applicability.

Real-time implementation is particularly demanding when MPC is deployed on embedded platforms with limited computational and memory resources. In such cases, it

is essential to utilize efficient solvers, exploit the problem structure, or apply suitable approximations to address strict time requirements while maintaining stability and satisfying system constraints.

A related challenge arises in the control of large-scale or network-structured systems, where centralized MPC may become impractical due to the high dimensionality of the optimization problem or is undesirable. Distributed and decentralized optimization techniques address the complexity problem of MPC by decomposing it into smaller subproblems that can be more handled more effectively. Nevertheless, these approaches introduce new complications, including the need for real-time coordination, convergence guarantees under finite iterations, and sensitivity to communication delays and packet loss in shared networks.

These challenges continue to be the subject of extensive research, with current work aiming to ensure real-time feasibility, improve computational efficiency, and develop scalable solutions for large-scale systems and embedded implementation also in the presence of constraints. The most relevant aspects for the successful real-time implementation are directly addressed in the contributions of this thesis.

2.2 Distributed Optimization

Distributed optimization refers to a class of methods designed to solve large-scale and structured optimization problems by decomposing them into smaller, more tractable subproblems. If a given problem exhibits a separable structure in its objective function or constraints, solving these subproblems independently (often in parallel) can significantly reduce overall computational effort. This makes distributed optimization particularly attractive for high-dimensional or time-critical applications where centralized computation becomes inefficient or impractical.

To present the structure of such problems, first, consider a equality-constrained optimization problem formulated as

$$\begin{aligned} \min_{z} \quad & f(z) \\ \text{s.t.} \quad & Dz = e \quad | \quad \lambda, \end{aligned} \tag{2.9}$$

where $z \in \mathbb{R}^n$ is the decision variable and $f : \mathbb{R}^n \to \mathbb{R}$ denotes the objective function. Coupling between decision variables is introduced via the affine equality constraints $Dz = e$, where $D \in \mathbb{R}^{m \times n}$ and $e \in \mathbb{R}^m$. The Lagrange multipliers associated with the equality constraints are denoted by $\lambda \in \mathbb{R}^m$.

If the optimization problem (2.9) admits a decomposable structure, it can be reformulated as

$$\begin{aligned} \min_{z} \quad & f(z) = \sum_{i=1}^{M} f_i(z_i) \\ \text{s.t.} \quad & Dz = \sum_{i=1}^{M} D_i z_i = e \quad | \quad \lambda, \end{aligned} \tag{2.10}$$

where $z_i \in \mathbb{R}^{n_z}$ are local decision variables associated with the i-th subproblem, and $f_i(z_i)$ represents the local contributions to the global objective function. The global coupling is captured by the aggregated equality constraint, where matrix $D = [D_1, \ldots, D_M]$ defines the distributed structure of the constraint coupling.

In general, distributed optimization algorithms are constructed by utilizing the Lagrangian function associated with the separable problem (2.10), which is given by

$$\mathcal{L}(z, \lambda) = \sum_{i=1}^{M} f_i(z_i) + \sum_{i=1}^{M} \lambda_i^\top D_i z_i. \tag{2.11}$$

The Lagrangian formulation is introduced because it preserves the optimal solution of the original problem (2.10) while presenting a separable structure that simplifies the decomposition into local subproblems of the form $\mathcal{L}_i(z_i, \lambda_i)$. The coordination across subsystems is achieved through iterative updates of the shared dual variables, ensuring consistency with the global coupling constraints.

Furthermore, if inequality constraints are imposed on the decision variables in the form $h(z) \leq 0$, and the constraint structure is separable, i.e., can be rewritten as $\sum_{i=0}^{M} h_i(z_i) \leq 0$, then these constraints are handled exclusively within the corresponding i-th local subproblem.

The key advantage of this formulation lies in its potential for parallelization. Each subproblem can be solved independently, often with significantly reduced computational complexity compared to the original problem formulation. The local solutions are then coordinated to recover a consistent global solution. This structure enables the application of iterative distributed algorithms, such as ADMM (Boyd et al., 2011) or ALADIN (Houska et al., 2016), which are particularly well-suited for large-scale and network-structured control systems.

2.2.1 ADMM Method

The Alternating Direction Method of Multipliers (ADMM) is a widely used first-order optimization algorithm, particularly well-suited for large-scale and distributed

convex optimization problems. It has been initially developed in Gabay and Mercier (1976) and later popularized through comprehensive theoretical analysis and broad applications in Boyd et al. (2011). The ADMM has become a standard tool used in process control, signal processing, and machine learning due to its good convergence properties, scalability, and ease of implementation.

ADMM is specifically designed for solving convex optimization problems with a separable structure, such as those presented in (2.10), where the global optimization problem can be decomposed into smaller, local subproblems coupled through linear equality constraints. The method builds upon the augmented Lagrangian, which extends the standard Lagrangian (2.11) by adding a quadratic penalty that enforces convergence to global optimum of (2.10). The augmented Lagrangian is defined as

$$\mathcal{L}_\rho(z, \bar{z}, \lambda) = \sum_{i=1}^{M} f_i(z_i) + \sum_{i=1}^{M} \lambda_i^\top D_i z_i + \frac{\rho}{2} \sum_{i=1}^{M} \|D_i(z_i - \bar{z}_i)\|_2^2, \tag{2.12}$$

where $\bar{z}$ represents a global consensus variable and $\rho > 0$ is the penalty parameter. At the optimal solution, the local variables z_i match the global consensus variable $\bar{z}$, causing the elimination of penalty term and thereby recovering the standard Lagrangian formulation. The introduction of the quadratic term improves convergence speed and numerical robustness compared with methods relying on standard Lagrangian function (i.e. dual decomposition method) as discussed in Boyd et al. (2011).

The ADMM algorithm operates iteratively by executing a sequences of local primal variable updates, and a global consensus and dual variable updates. In each iteration, the local subproblems are solved independently, typically in parallel, followed by a consensus or coordination step to align shared variables across the subsystems. The dual variables are then updated to enforce the satisfaction of the coupling constraints. This process is repeated until a stopping criterion is satisfied, which is commonly based on the norms of the primal and dual residuals.

Various formulations and extensions of ADMM have been developed in the literature, each offering different convergence guarantees and tailored to specific problem structures or application domains. In this work, we utilize a standard form of ADMM, adapted to the distributed MPC setting . The ADMM procedure for solving problem (2.10) is summarized in Algorithm 1. In Step 1), local subproblems are solved independently to compute local variable updates. These updates are then passed to Step 2), where coordination is achieved by solving a QP problem that enforces agreement among local solutions via global consensus variables. Subsequently, Step 3) updates the associated Lagrange multipliers based on the current primal residuals. This iterative process continues until the primal and dual optimality conditions specified in Step

4) are satisfied, thereby yielding a solution to the original distributed optimization problem (2.10) with additional inequality constraints.

Algorithm 1 ADMM Algorithm (Boyd et al., 2011)

Input: Set initial guesses $\bar{z}^1$, dual variable λ^1, penalty parameter $\rho > 0$, tolerance $\varepsilon > 0$ and iteration counter $\ell = 1$.

Repeat:

1) Solve local problems in parallel for each $i = \{1, \dots, M\}$

$$z_i^{\ell+1} := \arg\min_{z_i} \; f_i(z_i) + (\lambda_i^\ell)^\top D_i z_i + \frac{\rho}{2} \left\| D_i(z_i - \bar{z}_i^\ell) \right\|_2^2, \quad h_i(\bar{z}_i) \leq 0.$$

2) Evaluate global consensus step

$$\bar{z}^{\ell+1} := \arg\min_{\bar{z}} \; \frac{\rho}{2} \sum_{i=1}^{M} \left\| D_i(z_i^{\ell+1} - \bar{z}_i) \right\|_2^2 - (\lambda_i^\ell)^\top D_i \bar{z}_i.$$

3) Update dual variable $\lambda^{\ell+1} := \lambda^\ell + \rho \sum_{i=1}^{M} D_i(z_i^{\ell+1} - \bar{z}_i^\ell)$.

4) Terminate if $\left\| z^{\ell+1} - \bar{z}^{\ell+1} \right\| \leq \varepsilon$, $\left\| \lambda^{\ell+1} - \lambda^\ell \right\| \leq \varepsilon$.

5) Update $\ell \leftarrow \ell + 1$ and go to Step 1).

ADMM offers several advantages that make it particularly suitable for distributed control. Due to its simple algorithmic structure, low computational cost per iteration, and low communication needs, it is well suited for implementation on embedded platforms. Moreover, the ADMM algorithm can be completely decentralized (no need for centralized coordinator) due to the separable structure of the problem presented in Step 2) of Algorithm 1. This yields more robust implementation strategy, mainly for the network-structured systems that eliminates the one point of failure.

However, ADMM also presents certain limitations. Its convergence can be relatively slow when high solution accuracy is required, and its performance is sensitive to algorithmic parameters, most notably the penalty parameter ρ. Furthermore, ADMM lacks mechanisms for leveraging second-order information, which can limit its effectiveness in the presence of ill-conditioned problem structures or strong coupling between subsystems.

2.2.2 ALADIN Method

The Augmented Lagrangian-based Alternating Direction Inexact Newton method (ALADIN), initially introduced in Houska et al. (2016), is a second-order distributed

optimization algorithm specifically designed for structured and large-scale problems. ALADIN has attracted considerable interest due to its fast local convergence and theoretical guarantees for non-convex problems under relatively mild regularity assumptions. In contrast to first-order methods such as ADMM, ALADIN leverages second-order information by incorporating local Hessian approximations, resulting in improved convergence rates and solution accuracy, particularly in ill-conditioned or strongly coupled problem settings.

To solve the structured optimization problem introduced in (2.10), the ALADIN algorithm alternates between parallelizable local computations and a centralized coordination step. Similar to ADMM, ALADIN is based on the augmented Lagrangian framework. However, it introduces a modified quadratic penalty term to enhance flexibility. The augmented Lagrangian used in ALADIN is defined as

$$\mathcal{L}_\rho(z, \bar{z}, \lambda) = \sum_{i=1}^{M} f_i(z_i) + \sum_{i=1}^{M} \lambda^\top D_i z_i + \frac{\rho}{2} \sum_{i=1}^{M} \|z_i - \bar{z}_i\|_H^2, \tag{2.13}$$

where $\bar{z}_i$ denotes the reference (consensus) variable for subsystem i, and $\|\cdot\|_H^2$ represents a norm weighted by a symmetric positive definite matrix $H \succ 0$.

In this thesis, expressions of the form $\|\xi\|_E$ with $E \in \mathbb{S}_+^n \succ 0$, where $\mathbb{S}_+^n$ denotes the subspace of symmetric matrices in $\mathbb{R}^{n\times n}$, are used as shorthand for

$$\|\xi\|_E = \sqrt{\xi^\top E \xi},$$

even though this does not represent a typical matrix norm in the strict mathematical sense. This simplification is adopted for notational convenience.

Compared to ADMM, the scalar penalty parameter ρ in ALADIN is typically fixed (e.g., $\rho = 1$), making H the primary tuning parameter, which allows for variable-specific regularization.

As discussed in Houska et al. (2016), to ensure desirable convergence properties of the ALADIN algorithm, the local scaling matrix H should ideally be chosen as the exact Hessian of the local objective function, i.e., $H = \nabla^2 f(z)$, or a suitable positive definite approximation $H \approx \nabla^2 f(z)$. Using an accurate second-order approximation improves convergence speed and robustness, particularly in the presence of non-convexity.

In contrast to the well-established ADMM method, the ALADIN algorithm is relatively new and, although promising, has not been extensively explored in the literature. Nevertheless, several variations of ALADIN algorithmic formulations have been introduced over the years, tailored to different problem structures and application domains.

In Algorithm 2, we present the variant of ALADIN proposed in Jiang et al. (2021), which has been shown to improve convergence properties for distributed optimization problems.

Algorithm 2 consists of four primary steps. In Step 1), each decoupled (possibly nonlinear) subproblem based on the augmented Lagrangian formulation (2.14) is solved, preferably in parallel. Step 2) defines a stopping criterion based on primal and dual residuals. It can be shown that if the stopping conditions in Step 2) are satisfied, the local variable update z^ℓ in current iteration ℓ fulfills the first-order Karush-Kuhn-Tucker (KKT) conditions up to a term of order $\mathcal{O}(\varepsilon)$ for pre-defined tolerance ε (Houska and Jiang, 2021). In Step 3), the algorithm constructs a coupled equality-constrained QP problem (2.15) based on local solutions. Finally, in Step 4), the algorithm updates both the primal and dual variables based on the solution of the coupled QP. These updates are then used to initialize the next iteration. This procedure is repeated until convergence criteria are met.

It is worth to note that coupled QP (2.15) in Algorithm 2 can be solved in two different ways. In a centralized setup with coordination unit, the QP is solved using standard dense QP solvers. Alternatively, in a decentralized setup, where the goal is to eliminate reliance on a central coordinator, the QP can be solved through the communication and updates exchange directly between agents. Here, only neighbor-to-neighbor communication is required, as the resulting KKT system is sparse and can be efficiently handled using distributed numerical linear algebra techniques Engelmann et al. (2020).

ALADIN offers several advantages over traditional distributed optimization methods. Its convergence properties are well-established, such as linear convergence under convexity assumptions and superlinear convergence when additional regularity conditions, such as suitable Hessian matrix approximation and strict complementarity, are satisfied (Houska et al., 2016). In addition, ALADIN exhibits robustness to ill-conditioning and is capable of handling a broad class of linear and nonlinear constrained optimization problems, making it a compelling choice for complex control applications.

These benefits, however, come at the cost of increased computational complexity per iteration. In particular, ALADIN requires the solution of a centralized equality-constrained QP and the online evaluation of second-order derivative or its approximations. This additional computational overhead may limit the method's applicability in real-time implementations.

To summarize the key distinction, the most significant difference between the ADMM

Algorithm 2 ALADIN Algorithm (Jiang et al., 2021)

Input: Initial guess $\bar{z}^1$, dual variable λ^1, scaling matrices $H_i \succ 0$, tolerance $\varepsilon > 0$ and iteration counter $\ell = 1$.

Repeat:

1) Solve local problems in parallel for each $i = 1, \dots, M$

$$z_i^\ell := \arg\min_{z_i} \; f_i(z_i) + (\lambda_i^\ell)^\top D_i z_i + \frac{1}{2}\|z_i - \bar{z}_i^\ell\|_{H_i}^2, \quad h_i(z_i) \leq 0. \tag{2.14}$$

2) Terminate if $\left\|Dz^\ell - e\right\| \leq \varepsilon$, $\left\|z^\ell - \bar{z}^\ell\right\| \leq \varepsilon$.

3) Solve coupled QP

$$\min_{\bar{z}} \; \sum_{i=1}^{M} \frac{1}{2}\|\bar{z}_i - 2z_i^\ell + \bar{z}_i^\ell\|_{H_i}^2, \quad \text{s.t.} \quad \sum_{i=1}^{M} D_i \bar{z}_i = e \quad | \quad \Delta\lambda. \tag{2.15}$$

4) Update $\bar{z}^{\ell+1} \leftarrow \bar{z}$, $\lambda^{\ell+1} \leftarrow \lambda^\ell + \Delta\lambda$, $\ell \leftarrow \ell + 1$ and go to Step 1).

Algorithm 1 and the ALADIN Algorithm 2 lies in the mathematical formulation of the coordination step. By incorporating exact or approximate Hessian information, ALADIN achieves superior convergence properties. On the other hand, the structure of the coupled QPs in ALADIN makes decentralized implementation less straightforward. Therefore, the choice of a suitable algorithm depends on the specific problem structure and application context.

2.3 Distributed Model Predictive Control

Generally, the formulation of MPC problem (2.7) fits to the structure of the distributed optimization problem (2.10). Specifically, the objective function is naturally separable across the prediction horizon N, as are the inequality constraints. Moreover, the system dynamics, represented by equality constraints, introduces coupling between input and state variables across the prediction horizon.

To apply distributed optimization techniques to problem (2.7), we first define the associated Lagrangian function in a form that exposes the separable structure

$$\mathcal{L}(x, u, \lambda) = x_N^\top P x_N + \sum_{k=0}^{N-1} \left(x_k^\top Q x_k + u_k^\top R u_k\right) + \sum_{k=0}^{N-1} \lambda_k^\top (x_{k+1} - A x_k - B u_k), \tag{2.16}$$

where $\lambda = [\lambda_0^\top, \lambda_1^\top, \dots, \lambda_{N-1}^\top]^\top$ denotes the vector of Lagrange multipliers associated

with the system dynamics constraints.

By exploiting the separable structure of the MPC problem across the prediction horizon, the Lagrangian can be decomposed into $N+1$ functions, each corresponding to a single time step k of the prediction horizon, as follows

$$\mathcal{L}_0(u_0, \lambda_0) = u_0^\top R u_0 - \lambda_0^\top (A x_0 + B u_0), \tag{2.17a}$$
$$\mathcal{L}_k(x_k, u_k, \lambda_k, \lambda_{k-1}) = x_k^\top Q x_k + u_k^\top R u_k - \lambda_k^\top (A x_k + B u_k) + \lambda_{k-1}^\top x_k, \tag{2.17b}$$
$$\mathcal{L}_N(x_N, \lambda_{N-1}) = x_N^\top P x_N + \lambda_{N-1}^\top x_N, \tag{2.17c}$$

with $k \in \{1, \ldots, N-1\}$. In this decomposition, each function $\mathcal{L}_k$ contains only the state and input variables associated with a single k-th step of prediction horizon.

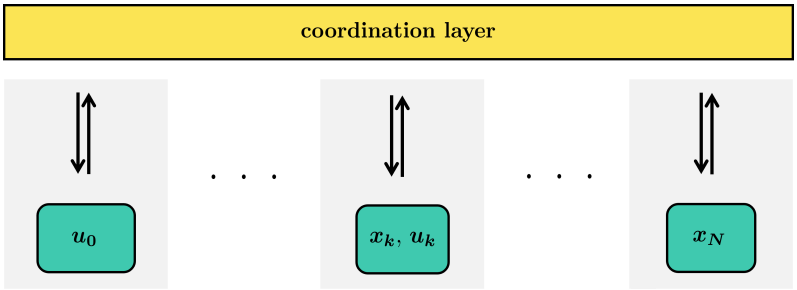


Figure 2.1: Visualization of the time-domain decomposition (across the prediction horizon) of the MPC problem. This leads to three distinct types of subproblem formulations.

Furthermore, provided that the constraint sets are also separable across the prediction horizon, i.e., $\mathbb{X} = \mathbb{X}_0 \times \ldots \times \mathbb{X}_{N-1}$ and $\mathbb{U} = \mathbb{U}_0 \times \ldots \times \mathbb{U}_{N-1}$, the original MPC problem (2.7) can be reformulated as $N+1$ independent subproblems in form

$$\min_{u_0 \in \mathbb{U}_0} \mathcal{L}_0(u_0, \lambda_0) + q(u_0 - \bar{u}_0), \tag{2.18a}$$
$$\min_{x_k \in \mathbb{X}_k, u_k \in \mathbb{U}_k} \mathcal{L}_k(x_k, u_k, \lambda_k, \lambda_{k-1}) + q(x_k - \bar{x}_k) + q(u_k - \bar{u}_k), \tag{2.18b}$$
$$\min_{x_N \in \mathbb{X}_N} \mathcal{L}_N(x_N, \lambda_{N-1}) + q(x_N - \bar{x}_N), \tag{2.18c}$$

for $k \in \{1, \ldots, N-1\}$, where the function $q(\cdot)$ denotes a quadratic penalty term that enforces coordination across the prediction horizon. The specific form of the consensus step, penalty term $q(\cdot)$, and the resulting coordination algorithm depend on the chosen strategy for handling distributed MPC problems. Simplified structure of resulting distributed MPC scheme is illustrated in Fig. 2.1.

In this thesis, we investigate the real-time implementation of distributed MPC using first- and second-order methods. The simplified diagram for solving distributed MPC problem by applying these methods is illustrated in Fig. 2.2. The iterative procedure is assumed to terminate once the stopping criterion is satisfied with an appropriate tolerance, resulting in a control action that is considered as optimal. The presented structure is well-suited for parallel execution of decoupled layer, which significantly reduces the computational burden and enables real-time applicability in distributed and embedded control settings.

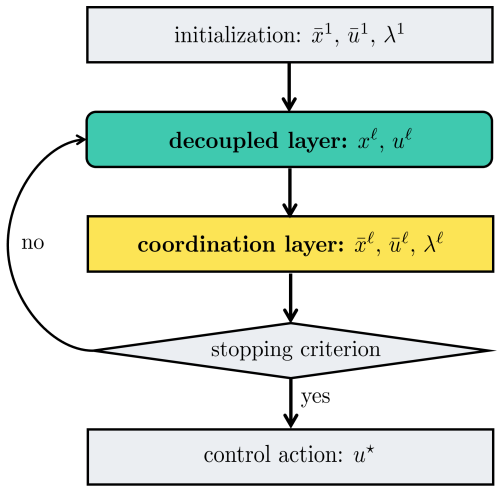


Figure 2.2: Visualization of the distributed MPC algorithm employing a stopping criterion that ensures the resulting control action is optimal (considering appropriate tolerance).

As a result, each k-th step of the MPC problem can be solved independently, enabling scalable and computationally efficient procedure. This is particularly beneficial considering the control of large-scale systems, where standard implementations may be computationally intractable. Several distributed MPC approaches have demonstrated promising results in this direction, including methods based on dual decomposition (Ferrari-Trecate et al., 2009), the ADMM (Zhang and Shi, 2011), and the ALADIN method (Engelmann et al., 2020).

Both ADMM and ALADIN serve as foundational components of the distributed optimization strategies proposed in this thesis. ADMM is particularly appealing for real-time MPC applications due to its guaranteed convergence for convex problems,

low per-iteration computational cost, and robustness under early termination, a crucial property for embedded control. Numerous studies have validated the effectiveness of ADMM in distributed and real-time MPC formulations (Ferrari-Trecate et al., 2009; Zhang and Shi, 2011), demonstrating its scalability and ease of implementation.

ALADIN, on the other hand, is especially well-suited for MPC problems with strong coupling or ill-conditioning, where second-order information can significantly improve convergence. Its distributed framework naturally aligns with the structure of MPC problems, and recent applications (Engelmann et al., 2020; Houska and Jiang, 2021) have verified its strong performance and scalability, even in real-time settings. The computational efficiency of the ALADIN method can be further improved by applying warm-starting strategies and/or early termination with theoretical guarantees.

Chapter 3

Generalized Stopping Criterion

In the context of real-time control, the computational burden associated with solving MPC problems to full optimality may exceed the available time limit, particularly when using embedded or resource-constrained hardware. This challenge is amplified when first-order optimization methods are employed, which, while scalable and computationally cheap per iteration, typically require many iterations to achieve high-precision solutions. Consequently, a fundamental trade-off arises between solution optimality and real-time feasibility. Suboptimal solutions, if computed rapidly, can offer a reasonable compromise, provided they preserve essential closed-loop properties such as feasibility and stability.

This chapter introduces a generalized stopping criterion for linear MPC problems with input constraints designed specifically to enable real-time implementation. The proposed approach enables the early termination of the underlying optimization algorithm (e.g., gradient-based or splitting methods) after a fixed number of iterations, resulting in a suboptimal control action. Crucially, the stopping condition is constructed to preserve theoretical guarantees, including the asymptotic stability of the closed-loop system. By enforcing a fixed number of solver iterations, the resulting computational load becomes predictable and bounded. This represents an essential feature for real-time applications with strict timing requirements. The results of this chapter summarize and extend our peer-reviewed contribution in Fedorová et al. (2023).

The subsequent sections detail the formulation of the generalized stopping criterion, its integration into first-order methods such as projected gradient and ADMM, and demonstrate its effectiveness through simulation-based evaluation. This theoretical contribution lays the foundation for scalable and certifiable real-time MPC strategies, which are introduced in the following chapters.

3.1 Linear MPC with Input Constraints

We consider the following standard linear MPC formulation with only input constraints as follows

$$V(x(t)) = \min_{x,u} x_N^\top P x_N + \sum_{k=0}^{N-1} \left(x_k^\top Q x_k + u_k^\top R u_k\right) \tag{3.1a}$$

$$\text{s.t.} \quad x_{k+1} = Ax_k + Bu_k, \quad k \in \{0, \dots, N-1\}, \tag{3.1b}$$

$$u_k \in \mathbb{U}_k, \quad k \in \{0, \dots, N-1\}, \tag{3.1c}$$

$$x_0 = x(t), \tag{3.1d}$$

where $V(x(t))$ denotes the value function associated with the MPC problem. We now introduce the following assumptions to ensure the desired theoretical properties.

Assumption 2. *We assume that the following conditions hold:*

- *The matrix pair (A, B) is controllable.*
- *The weighting matrices Q, R, and P are symmetric and positive definite.*
- *The terminal cost matrix P is the solution of the discrete-time algebraic Riccati equation* (2.8).
- *The input constraint sets $\mathbb{U}_k$ are convex, closed, and contain the origin in their strict interiors.*

Assumption 2 ensures that the MPC problem in (3.1) is strongly convex with a unique global minimizer. Since the formulation does not include state constraints, recursive feasibility is inherently satisfied.

Assumption 3. *The set of feasible initial conditions of the MPC problem* (3.1) *is either rigorously contained within the corresponding region of attraction or is precisely aligned with this set. This property is a direct consequence of introducing the terminal penalty in* (3.1)*, which is specifically designed to satisfy the conditions of Theorem 1 in Limon et al. (2006).*

Under the assumption that $V(x(t))$ serves as a Lyapunov function, and provided that Assumptions 2–3 are satisfied, the optimal MPC control law $\mu(x) = u^\star(x(t))$ guarantees the standard decrease condition presented in Rawlings et al. (2017) in following form

$$V(Ax(t) + B\mu(x)) \leq V(x(t)) - \left(x(t)^\top Q x(t) + \mu(x)^\top R \mu(x)\right), \tag{3.2}$$

which ensures asymptotic stability of the closed-loop system under the optimal receding horizon policy.

Proposition 1 (Proposition 2.16 in Rawlings et al. (2009))**.** *Let Assumptions 2 and 3 hold. Then the MPC controller defined by $u^\star(x(t))$ ensures asymptotic stability, i.e.,*

$$V(Ax(t) + Bu^\star(x(t))) \leq V(x(t)) - \left(\|x(t)\|_Q^2 + \|u^\star(x(t))\|_R^2\right). \tag{3.3}$$

In practice, however, computing the exact optimal control input $u^\star(x(t))$ at each control step may be computationally intractable, especially on embedded platforms. This is particularly true for first-order optimization methods, which are computationally efficient per iteration but typically require many iterations to achieve high-accuracy solutions.

Consequently, it is often necessary to terminate the solver early after a fixed number of iterations, yielding a suboptimal control action. In such cases, the inequality (3.2) may no longer hold, which could compromise closed-loop stability.

To address this issue, we propose a generalized stopping criterion that is compatible with any first-order algorithm exhibiting linear convergence. The criterion defines a fixed number of iterations sufficient to guarantee that the resulting suboptimal control law still ensures the Lyapunov descent property (3.2), thereby preserving closed-loop stability even in the presence of early termination.

3.2 Criterion for Fixed Number of Iterations

To derive the formula for the generalized stopping criterion, we begin by defining a suboptimal control law. Let $u^{\bar{\ell}}(x(t))$ denote the suboptimal input obtained after $\bar{\ell}$ iterations of any iterative first-order optimization method. Applying this control action to the LTI system (2.2) yields the predicted successor state

$$\tilde{x}(t + T_\mathrm{s}) = Ax(t) + Bu^{\bar{\ell}}(x(t)), \tag{3.4}$$

which may differ from the optimal successor state obtained with the optimal control input $u^\star(x(t))$ as

$$\tilde{x}(t + T_\mathrm{s}) \neq x(t + T_\mathrm{s}) = Ax(t) + Bu^\star(x(t)). \tag{3.5}$$

To simplify the notation, we will use $\tilde{x}^+ = \tilde{x}(t + T_\mathrm{s})$ and $x^+ = x(t + T_\mathrm{s})$ throughout the remainder of the thesis.

To facilitate the derivation of a generalized stopping criterion, we impose the following structural properties on the states and optimal control action.

Proposition 2. *There exists a neighborhood $\mathcal{B}_\delta^{n_x}$ of the origin such that, for all $x \in \mathcal{B}_\delta^{n_x}$, the optimal solution to (3.1) lies strictly within the interior of the input constraint sets $\mathbb{U}_k$ for all $k \in \{0, \dots, N-1\}$.*

Assumption 4. *There exists a real-valued constant $\gamma > 0$ such that*

$$\|u^\star(x(t))\|_2 \leq \gamma \|x(t)\|_Q. \tag{3.6}$$

As shown in Borrelli et al. (2003), the optimal control input $u^\star(x(t))$ is a piecewise affine function of the state $x(t)$. Therefore, the constant γ in (3.6) can be computed offline prior to the real-time implementation of the MPC controller.

Assumption 5. *Assume that the convex set $\mathbb{Q} = \{x(t) \in \mathbb{R}^{n_x} \mid \|x(t)\|_Q \leq 1\}$ is non-empty and fully contained within the neighborhood of the origin $\mathcal{B}_\delta^{n_x}$.*

Note that Assumption 5 can be satisfied by appropriately scaling the weighting matrices Q and R in the MPC formulation (3.1), as illustrated in Fig. 3.1.

The following theorem provides conditions under which the closed-loop system remains asymptotically stable when a control input is computed using a first-order optimization method with a fixed number of iterations applied to the MPC problem (3.1).

Theorem 3. *Let Assumptions 2–5 hold, and let the MPC problem (3.1) be feasible. Let $\kappa < 1$ denote the linear convergence rate of the employed optimization algorithm. Furthermore, there exist constants $\eta_1 \geq 0$ and $\eta_2 > 0$ such that*

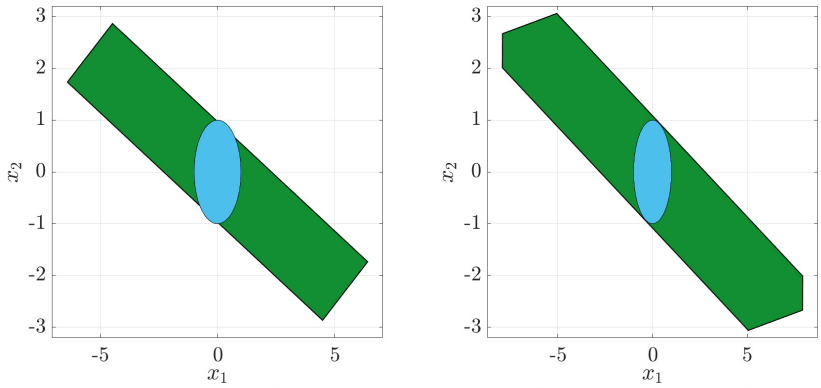
$$|V(\tilde{x}^+) - V(x^+)| \leq \eta_1 \|\tilde{x}^+ - x^+\|_2 + \frac{\eta_2}{2} \|\tilde{x}^+ - x^+\|_2^2 \tag{3.7}$$

hold. If the fixed number of iterations $\bar{\ell} \in \mathbb{Z}$ satisfies

$$\bar{\ell} > \frac{\log(2\bar{\eta}_1\gamma + 2\bar{\eta}_2\gamma^2)}{\log(1/\kappa)}, \tag{3.8}$$

with $\bar{\eta}_1 = f_{\eta_1}(\eta_1)$ and $\bar{\eta}_2 = f_{\eta_2}(\eta_2)$, then applying the suboptimal control input $u(t) = u^{\bar{\ell}}(x(t))$ to the LTI system (2.2) guarantees asymptotic stability of the resulting closed-loop system.

Proof. We begin by rewriting the inequality (3.7), which expresses an upper bound on the difference in value functions between the optimal and suboptimal state trajectories,



a) Assumption 5 violated. b) Assumption 5 holds.

Figure 3.1: Visualization of the LQR-based control invariant set (green) for two different configurations of the weighting matrices (Q, R) in the MPC design and set $\mathbb{Q}$ (blue) defined in Assumption 5: a) the set $\mathbb{Q}$ is not fully contained within the LQR-based control invariant set, b) the LQR-based invariant set entirely contains $\mathbb{Q}$.

using the system model (2.2), as follows

$$|V(\tilde{x}^+) - V(x^+)| \leq \eta_1 \|Ax(t) + B u_0^{\bar{\ell}} - (Ax(t) + B u_0^\star)\|_2 + \frac{\eta_2}{2} \|Ax(t) + B u_0^{\bar{\ell}} - (Ax(t) + B u_0^\star)\|_2^2 \tag{3.9}$$

$$= \eta_1 \|B(u_0^{\bar{\ell}} - u_0^\star)\|_2 + \frac{\eta_2}{2} \|B(u_0^{\bar{\ell}} - u_0^\star)\|_2^2, \tag{3.10}$$

where $u_0^\star$ and $u_0^{\bar{\ell}}$ denote the first elements of the input sequences $u^\star = [u_0^{\star\top}, \ldots, u_{N-1}^{\star\top}]^\top$ and $u^{\bar{\ell}} = [u_0^{\bar{\ell}\top}, \ldots, u_{N-1}^{\bar{\ell}\top}]^\top$, respectively. To facilitate the analysis, we define the matrix $\tilde{B} = [B,\ 0,\ \ldots,\ 0] \in \mathbb{R}^{n_\mathrm{x} \times N n_\mathrm{u}}$, which maps the full decision vector u to the control input u_0. Using this definition, we can rewrite the following terms from (3.7) as

$$B(u_0^{\bar{\ell}} - u_0^\star) = \tilde{B}(u^{\bar{\ell}} - u^\star). \tag{3.11}$$

Now assume that the employed first-order optimization algorithm converges linearly with rate $\kappa < 1$. Then, for any iteration $\ell \in \mathbb{Z}$ of the algorithm, the following inequality holds

$$\|u^{\ell+1} - u^\star\|_2 \leq \kappa \|u^\ell - u^\star\|_2, \tag{3.12}$$

which implies

$$\|u^\ell - u^\star\|_2 \leq \kappa^\ell \|u^0 - u^\star\|_2. \tag{3.13}$$

Substituting (3.11) and (3.13) into (3.10), we obtain

$$\begin{aligned} |V(\tilde{x}^+) - V(x^+)| &\leq \bar{\eta}_1 \|u^{\bar{\ell}} - u^\star\|_2 + \bar{\eta}_2 \|u^{\bar{\ell}} - u^\star\|_2^2 \\ &\leq \bar{\eta}_1 \kappa^{\bar{\ell}} \|u^0 - u^\star\|_2 + \bar{\eta}_2 \kappa^{2\bar{\ell}} \|u^0 - u^\star\|_2^2, \end{aligned} \tag{3.14}$$

where we define

$$\bar{\eta}_1 = \eta_1 \sqrt{\nu_{\max}(\tilde{B}^\top \tilde{B})}, \quad \bar{\eta}_2 = \frac{\eta_2}{2} \nu_{\max}(\tilde{B}^\top \tilde{B}), \tag{3.15}$$

with $\nu_{\max}(\cdot)$ denoting the largest eigenvalue of a given symmetric matrix.

Finally, using Assumption 4, which provides the bound $\|u^\star\|_2 \leq \gamma \|x(t)\|_Q$ and assuming $\|u^0 - u^\star\|_2 \leq 2\gamma \|x(t)\|_Q$, which can be enforced by initial rescale of input variable in the first-order algorithm, we obtain the upper bound in the form

$$|V(\tilde{x}^+) - V(x^+)| \leq 2\bar{\eta}_1 \gamma \kappa^{\bar{\ell}} \|x(t)\|_Q + 2\bar{\eta}_2 \gamma^2 \kappa^{2\bar{\ell}} \|x(t)\|_Q^2. \tag{3.16}$$

This bound quantifies the maximum deviation in the value function between the optimal state trajectory and the suboptimal one, where the latter results from applying a suboptimal control input after a finite number of solver iterations.

We now proceed to establish the stability guarantee in two steps based on the value of state measurement $x(t)$. First, under the condition $\|x(t)\|_Q > 1$, we have

$$\|x(t)\|_Q^2 \geq \|x(t)\|_Q, \tag{3.17}$$

and since $\kappa < 1$, it follows that $\kappa^{\bar{\ell}} > \kappa^{2\bar{\ell}}$. Consequently, the upper bound in (3.16) can be simplified as

$$|V(\tilde{x}^+) - V(x^+)| \leq (2\bar{\eta}_1 \gamma + 2\bar{\eta}_2 \gamma^2) \kappa^{\bar{\ell}} \|x(t)\|_Q^2 = \beta_{>1} \|x(t)\|_Q^2, \tag{3.18}$$

where

$$\beta_{>1} = (2\bar{\eta}_1 \gamma + 2\bar{\eta}_2 \gamma^2) \kappa^{\bar{\ell}}. \tag{3.19}$$

In order to establish asymptotic stability of the closed-loop system under suboptimal control action, we need to ensure that the Lyapunov descent condition (3.2) still holds. Therefore, we algebraically extend (3.2) by adding $V(\tilde{x}^+)$ to both sides resulting in

$$V(x^+) + V(\tilde{x}^+) \leq V(x(t)) + V(\tilde{x}^+) - \|x(t)\|_Q^2, \tag{3.20}$$

which can be further rearranged as

$$V(\tilde{x}^+) \leq V(x(t)) + (V(\tilde{x}^+) - V(x^+)) - \|x(t)\|_Q^2. \tag{3.21}$$

If we can show that the term $V(\tilde{x}^+) - V(x^+) - \|x(t)\|_Q^2$ is negative, we can guarantee that the value function continues to decrease despite the application of a suboptimal control input. This, in turn, ensures asymptotic stability even when the underlying optimization problem is terminated early. To establish these properties, we substitute the upper bound derived in (3.18) into (3.21), yielding

$$V(\tilde{x}^+) \leq V(x(t)) + \beta_{>1}\|x(t)\|_Q^2 - \|x(t)\|_Q^2 = V(x(t)) - (1-\beta_{>1})\|x(t)\|_Q^2. \quad (3.22)$$

As mentioned, to ensure the asymptotic stability of the closed-loop system under a suboptimal control law, we need to enforce a strict decrease in the value function. This requirement translates to the condition $(1-\beta_{>1}) > 0$, or equivalently, $\beta_{>1} < 1$. Substituting the definition of $\beta_{>1}$ leads to the inequality

$$(2\bar{\eta}_1\gamma + 2\bar{\eta}_2\gamma^2)\,\kappa^{\bar{\ell}} < 1, \quad (3.23)$$

which is used to derive a lower bound on the number of iterations $\bar{\ell}$. Solving this inequality for $\bar{\ell}$ as

$$\log\left((2\bar{\eta}_1\gamma + 2\bar{\eta}_2\gamma^2)\kappa^{\bar{\ell}}\right) < \log(1), \quad (3.24a)$$

$$\log(2\bar{\eta}_1\gamma + 2\bar{\eta}_2\gamma^2) + \bar{\ell}\log(\kappa) < 0, \quad (3.24b)$$

$$\bar{\ell}\log(\kappa) < -\log(2\bar{\eta}_1\gamma + 2\bar{\eta}_2\gamma^2), \quad (3.24c)$$

$$\bar{\ell} > \frac{-\log(2\bar{\eta}_1\gamma + 2\bar{\eta}_2\gamma^2)}{\log(\kappa)}, \quad (3.24d)$$

yields the minimum number of iterations required to theoretically guarantee closed-loop stability. Note that mathematical operations applied in (3.24) rely on the assumption $\kappa < 1$, which implies $\log(\kappa) < 0$. The final formula can then be equivalently written as

$$\bar{\ell} > \frac{\log(2\bar{\eta}_1\gamma + 2\bar{\eta}_2\gamma^2)}{\log(1/\kappa)}. \quad (3.25)$$

This inequality defines the minimum number of iterations required for the Lyapunov descent condition (3.22) to hold despite suboptimality. As a result, the closed-loop system remains asymptotically stable when control actions are computed by terminating the optimization algorithm after $\bar{\ell}$ iterations, under the assumption that $\|x(t)\|_Q > 1$.

However, if $\|x(t)\|_Q \leq 1$, the previous derivation no longer applies directly. In this case, the quadratic term $\|x(t)\|_Q^2$ is dominated by the linear term $\|x(t)\|_Q$, which complicates the derivation of a suitable upper bound. In particular, substituting the bound from (3.18) would not preserve the form of the Lyapunov descent condition (3.21).

To address this issue, we invoke Proposition 2, which guarantees the existence of a neighborhood $\mathcal{B}_\delta^{n_x}$ of the origin in which the input constraints are inactive. In this

region, the value function reduces to the unconstrained quadratic form $V(x) = x^\top Px$. As a result, the first-order term in the upper bound is no longer present, i.e., $\eta_1 = 0$, and inequality (3.7) simplifies to

$$|V(\tilde{x}^+) - V(x^+)| \leq \frac{\eta_2}{2}\|\tilde{x}^+ - x^+\|_2^2. \tag{3.26}$$

Furthermore, the $\{x(t) : \|x(t)\|_Q \leq 1\}$ is fully contained within $\mathcal{B}_\delta^{n_x}$ as defined in Assumption 5. Thus, the simplified bound (3.26) applies throughout this region. Following the same derivation as in the previous case leads to the inequality

$$|V(\tilde{x}^+) - V(x^+)| \leq 2\bar{\eta}_2\gamma^2\kappa^{2\bar{\ell}}\|x(t)\|_Q^2 = \beta_{\leq 0}\|x(t)\|_Q^2. \tag{3.27}$$

Substituting this bound into the modified Lyapunov descent condition (3.21) yields

$$V(\tilde{x}^+) \leq V(x(t)) - (1 - \beta_{\leq 0})\|x(t)\|_Q^2, \tag{3.28}$$

which guarantees asymptotic stability of the closed-loop system provided that $1-\beta_{\leq 0} > 0$. This condition holds if the number of solver iterations $\bar{\ell}$ satisfies the following inequality

$$\begin{aligned}
\beta_{\leq 0} = 2\bar{\eta}_2\gamma^2\kappa^{2\bar{\ell}} &< 1, && \text{(3.29a)}\\
\log(2\bar{\eta}_2\gamma^2\kappa^{2\bar{\ell}}) &< \log(1), && \text{(3.29b)}\\
\log(2\bar{\eta}_2\gamma^2) + \log(\kappa^{2\bar{\ell}}) &< 0, && \text{(3.29c)}\\
2\bar{\ell}\log(\kappa) &< -\log(2\bar{\eta}_2\gamma^2), && \text{(3.29d)}\\
\bar{\ell} &> \frac{-\log(2\bar{\eta}_2\gamma^2)}{2\log(\kappa)}, && \text{(3.29e)}\\
\bar{\ell} &> \frac{\log(2\bar{\eta}_2\gamma^2)}{2\log(1/\kappa)}. && \text{(3.29f)}
\end{aligned}$$

This expression defines the minimum number of iterations required to maintain asymptotic stability in the case where $\|x(t)\|_Q \leq 1$. Consequently, we can consider two implementation strategies. The first approach involves evaluating the norm $\|x(t)\|_Q$ at each control step and, depending on whether the condition $\|x(t)\|_Q \leq 1$ is satisfied, selecting the corresponding pre-computed iteration bound. This results in a state-dependent switching mechanism for the number of iterations in the real-time optimization routine, as outlined in Algorithm 3.

Algorithm 3 Fixed-Iteration Real-Time MPC with State-Dependent Criterion

Offline: Precompute iteration bounds $\bar{\ell}_{\leq 1} \in \mathbb{Z}$, $\bar{\ell}_{>1} \in \mathbb{Z}$ such that

$$\bar{\ell}_{\leq 1} = \left\lceil \frac{\log(2\bar{\eta}_2\gamma^2)}{2\log(1/\kappa)} \right\rceil, \quad \bar{\ell}_{>1} = \left\lceil \frac{\log(2\bar{\eta}_1\gamma + 2\bar{\eta}_2\gamma^2)}{\log(1/\kappa)} \right\rceil$$

Online:

1) Obtain state measurement $x(t)$
2) Set the value of $\bar{\ell}$ as

$$\bar{\ell} = \begin{cases} \bar{\ell}_{\leq 1}, & \text{if } \|x(t)\|_Q \leq 1 \\ \bar{\ell}_{>1}, & \text{otherwise} \end{cases}$$

3) Run first-order optimization algorithm (for reference see Section 3.3) with $\bar{\ell}$ iterations to solve MPC problem
4) Go to the Step 1)

Alternatively, we can eliminate the dependence of the required number of iterations on the current state measurement by identifying a single conservative bound that ensures stability for all admissible states. In this way, the online evaluation of the state-dependent condition is eliminated, simplifying the real-time implementation.

To this end, we compare the two fixed-iteration formulas derived for the cases $\|x(t)\|_Q \leq 1$ and $\|x(t)\|_Q > 1$, denoted by $\bar{\ell}_{\leq 1}$ and $\bar{\ell}_{>1}$, respectively. If we can show that $\bar{\ell}_{>1}$ is always equal to or greater than $\bar{\ell}_{\leq 1}$, then the single formula $\bar{\ell}_{>1}$ can be used uniformly in each control step, regardless of the state measurement. Let us assume that

$$\bar{\ell}_{>1} > \bar{\ell}_{\leq 1} \tag{3.30a}$$

$$\frac{\log(2\bar{\eta}_1\gamma + 2\bar{\eta}_2\gamma^2)}{\log(1/\kappa)} > \frac{\log(2\bar{\eta}_2\gamma^2)}{2\log(1/\kappa)} \tag{3.30b}$$

$$\log(2\bar{\eta}_1\gamma + 2\bar{\eta}_2\gamma^2) > \frac{1}{2}\log(2\bar{\eta}_2\gamma^2) \tag{3.30c}$$

$$2\log(2\bar{\eta}_1\gamma + 2\bar{\eta}_2\gamma^2) > \log(2\bar{\eta}_2\gamma^2), \tag{3.30d}$$

which holds due to the monotonicity of the logarithmic function and the fact that all parameters involved are strictly positive by definition. This leads to a unified conservative bound for the fixed number of iterations,

$$\bar{\ell} > \frac{\log(2\bar{\eta}_1\gamma + 2\bar{\eta}_2\gamma^2)}{\log(1/\kappa)} > \frac{\log(2\bar{\eta}_2\gamma^2)}{2\log(1/\kappa)}, \tag{3.31}$$

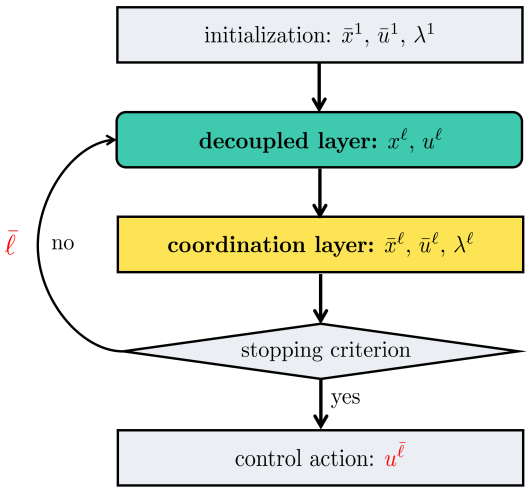


Figure 3.2: Visualization of the distributed MPC algorithm employing a stopping criterion if the form of a fixed number of iterations provides suboptimal control action.

which ensures the asymptotic stability guarantee of the closed-loop system where the suboptimal control input $u_0^{\bar{\ell}}$ is applied to the LTI system (2.2). □

We emphasize that by upper bounding the original Lyapunov descent condition (3.21) using (3.18), we obtained the conservative control performance. As a result, a smaller number of iterations may still be sufficient to ensure the asymptotic stability of the closed-loop system.

Additionally, it is important to mention that computing the constants η_1 and η_2, which are required for the stopping criterion, can be challenging in practice. For small-scale systems, these constants can often be determined using techniques from explicit MPC theory. However, in the case of larger or more complex systems, deriving explicit solutions becomes computationally infeasible. In such situations, conservative estimates must be used instead. One viable approach for obtaining such bounds is based on approximate dynamic programming, as proposed in Wang et al. (2012).

Finally, we now utilize the derived formula for $\bar{\ell}$ to update the structure of the MPC algorithm shown in Fig. 2.2, replacing it with a version that performs only a fixed number of iterations and producing suboptimal control action. As proven, even when

suboptimal control actions generated by the MPC algorithm visualized in Fig. 3.2 are applied to the controlled system, the closed-loop system remains asymptotically stable. As a result, Fig. 3.2 illustrates the simplified scheme of the suboptimal MPC algorithm used throughout this thesis.

3.3 First-Order Methods

In this section, we introduce two widely used first-order methods for solving linear MPC problems with input constraints. To facilitate their implementation, we first reformulate the original MPC problem (3.1) into its dense QP form in the following way

$$\begin{aligned} V(x(t)) = \min_{u} \quad & J(u, x(t)) = \frac{1}{2} \begin{bmatrix} u \\ x(t) \end{bmatrix}^\top \begin{bmatrix} \mathbf{H} & \mathbf{S} \\ \mathbf{S}^\top & \mathbf{D} \end{bmatrix} \begin{bmatrix} u \\ x(t) \end{bmatrix} \\ \text{s.t.} \quad & u \in \mathbb{U} = \mathbb{U}_0 \times \cdots \times \mathbb{U}_{N-1}, \end{aligned} \tag{3.32}$$

where $\mathbb{U} \subset \mathbb{R}^{Nn_\mathrm{u}}$ denotes the set of admissible input sequences, the matrices $\mathbf{S} \in \mathbb{R}^{n_\mathrm{x} \times Nn_\mathrm{u}}$ and $\mathbf{D} \in \mathbb{R}^{n_\mathrm{x} \times n_\mathrm{x}}$ are given by $\mathbf{S} = \tilde{\Psi}^\top \widetilde{Q} \tilde{\Phi}, \mathbf{D} = \tilde{\Psi}^\top \widetilde{Q} \tilde{\Psi}$, where $\tilde{\Psi}^\top = [A, A^2, \ldots, A^N]^\top$ and $\tilde{\Phi}_k = [A^{k-1}B, A^{k-2}B, \ldots, B, 0, \ldots, 0] \in \mathbb{R}^{n_\mathrm{x} \times Nn_\mathrm{u}}$. The symmetric positive definite matrix $\mathbf{H} \in \mathbb{S}^{Nn_\mathrm{u}}_{++}$ is defined by

$$\mathbf{H} = \underbrace{\begin{bmatrix} R & & \\ & \ddots & \\ & & R \end{bmatrix}}_{\widetilde{R}} + \begin{bmatrix} \tilde{\Phi}_1 \\ \vdots \\ \tilde{\Phi}_N \end{bmatrix}^\top \underbrace{\begin{bmatrix} Q & & \\ & \ddots & \\ & & P \end{bmatrix}}_{\widetilde{Q}} \underbrace{\begin{bmatrix} \tilde{\Phi}_1 \\ \vdots \\ \tilde{\Phi}_N \end{bmatrix}}_{\tilde{\Phi}}, \tag{3.33}$$

The QP formulation (3.32) is mathematically equivalent to the original MPC problem (3.1), and all theoretical results developed in the context of the generalized stopping criterion remain valid. However, the numerical simulations and algorithmic evaluations presented in the following sections are conducted on this condensed formulation, which is more suitable for implementing proposed first-order methods.

Here, we present two first-order methods: the projected gradient descent and the ADMM, both of which are employed to validate the proposed generalized stopping criterion. It is important to note that the ADMM implementation presented in this section corresponds to the most basic variant of the algorithm. This simplified version is used only for validation purposes of the general stopping criterion. In contrast, the version introduced earlier in Section 2.2.1 incorporates the distribution and parallelization of the local layer to improve computational efficiency. While both variants are based on the same underlying mathematical principles, the version

with parallelization of the local layer, as discussed in the theoretical background in Section 2.2.1, is better suited for large-scale and distributed applications.

- **Projected Gradient Descent Method**

  The Projected Gradient Descent Method (PGDM) is a straightforward extension of the classical gradient descent algorithm for constrained optimization problems. In the context of solving the QP formulation (3.32) of the MPC problem, the method iteratively updates the decision variable u by taking a gradient step followed by a projection onto the input-constrained set $\mathbb{U}$

  $$u^{\ell+1} = \mathrm{Proj}_{\mathbb{U}}\left(u^{\ell} - \tau \nabla J(u^{\ell}, x(t))\right), \tag{3.34}$$

  where $u^{\ell} \in \mathbb{R}^{Nn_{\mathrm{u}}}$ denotes the current iterate, $\tau > 0$ is a fixed step size, and $\mathrm{Proj}_{\mathbb{U}}(\cdot)$ denotes the Euclidean projection onto the feasible set $\mathbb{U}$.

  Assuming the cost function $J(u, x(t))$ is μ-strongly convex and L-smooth, which holds when the matrix $\mathbf{H}$ in (3.32) satisfies $0 < \mu\mathbf{I} \preceq \mathbf{H} \preceq L\mathbf{I}$, the convergence of PGDM is guaranteed. Specifically, setting the step size $\tau = \frac{1}{L}$ results in a linear convergence rate given by

  $$\kappa = 1 - \frac{\mu}{L}, \tag{3.35}$$

  where $\kappa < 1$ ensures geometric decay of the suboptimality gap. For further theoretical background, see Bubeck (2015).

- **Alternating Direction Method of Multipliers**

  The basic variant of the ADMM method applied to the QP formulation (3.32) of the MPC problem has the following steps

  $$u^{\ell+1} = \arg\min_{u} \; J(u, x(t)) + u^{\top}\lambda^{\ell} + \frac{\rho}{2}\|u - \bar{u}^{\ell}\|_2^2, \tag{3.36a}$$

  $$\bar{u}^{\ell+1} = \arg\min_{v \in \mathbb{U}} \; -\bar{u}^{\top}\lambda^{\ell} + \frac{\rho}{2}\|u^{\ell+1} - \bar{u}\|_2^2, \tag{3.36b}$$

  $$\lambda^{\ell+1} = \lambda^{\ell} + \rho(u^{\ell+1} - \bar{u}^{\ell+1}), \tag{3.36c}$$

  where $u^{\ell} \in \mathbb{R}^{Nn_{\mathrm{u}}}$ and $\bar{u}^{\ell} \in \mathbb{R}^{Nn_{\mathrm{u}}}$ denote the primal and coordination variables, $\lambda^{\ell} \in \mathbb{R}^{Nn_{\mathrm{u}}}$ is the dual variable (Lagrange multiplier), and $\rho > 0$ is a user-defined penalty parameter.

  The convergence rate of ADMM, when applied to QPs, has been analyzed in Ghadimi et al. (2015). Under suitable regularity assumptions, the method achieves linear convergence with rate

  $$\kappa = \frac{1}{2}\|2\mathbf{M} - \mathbf{I}\|, \tag{3.37}$$

where the matrix $\mathbf{M}$ is defined by

$$\mathbf{M} = \rho \mathbf{G}\mathbf{H}^{-1}\mathbf{G}^\top - \rho \mathbf{G}\mathbf{H}^{-1}\mathbf{G}^\top (I + \rho \mathbf{G}\mathbf{H}^{-1}\mathbf{G}^\top)^{-1} \rho \mathbf{G}\mathbf{H}^{-1}\mathbf{G}^\top. \tag{3.38}$$

Here, $\mathbf{H}$ is the Hessian matrix from (3.32), and $\mathbf{G} \in \mathbb{R}^{2Nn_\mathrm{u} \times Nn_\mathrm{u}}$ defines the input constraint set $\mathbb{U} = \{u \in \mathbb{R}^{n_\mathrm{u}} \mid \mathbf{G}u \leq w \text{ with } w \in \mathbb{R}^{2Nn_\mathrm{u}}$.

Generally, the generalized stopping criterion can be applied to any first-order optimization method as long as it exhibits linear convergence. The subsequent numerical simulations demonstrate that our proposed criterion provides a substantial reduction in the number of solver iterations required while reporting only minimal suboptimality in the resulting closed-loop performance.

3.4 Numerical Case Study

To evaluate the properties of the proposed stopping criterion, we consider two benchmark systems. The first is a double integrator, which serves as a simple system that allows for detailed analysis of the stopping conditions. The second is a spring-damper system, representing a more complex dynamic setup. We demonstrate that the proposed stopping criterion is applicable regardless of system complexity and can significantly reduce computational time at the cost of a small degree of suboptimality. All simulations were performed in `MATLAB R2022b`.

Furthermore, the PGDM and ADMM algorithms were implemented in `MATLAB`, as described in Section 3.3. These implementations do not rely on external solvers or any optimization toolboxes. To evaluate the effectiveness of the proposed generalized stopping criterion, we consider five configurations: (i) nominal MPC solved via `GUROBI`, (ii) PGDM with a standard tolerance-based stopping criterion, (iii) PGDM with the generalized stopping criterion, denoted PGDM($\bar{\ell}$), (iv) ADMM with the standard stopping criterion, and (v) ADMM with the generalized criterion, denoted ADMM($\bar{\ell}$). The standard tolerance-based stopping condition is defined as $|\Delta J| < \varepsilon$, with $\varepsilon = 10^{-6}$.

3.4.1 Double Integrator System

To analyze the properties of the proposed stopping criterion, we adopt the well-known double integrator system, characterized by the system matrices

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 0.5 \end{bmatrix}, \quad B = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}.$$

The model predictive controller defined in (3.32) is designed using the following weight matrices

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad P = \begin{bmatrix} 2.367 & 1.118 \\ 1.118 & 2.588 \end{bmatrix}, \quad R = 1,$$

where P is computed as the solution to the discrete-time Riccati equation (2.8). The input constraint set is defined as $\mathbb{U} = \{u \in \mathbb{R} \mid -1 \leq u \leq 1\}$, and the finite prediction horizon is set to $N = 10$.

The parameters used to compute the convergence rates κ and the corresponding fixed number of iterations $\bar{\ell}$ are provided in Table 3.1. The values of κ_{PGDM} and κ_{ADMM} are evaluated using (3.35) and (3.37), respectively. The corresponding fixed iteration bounds $\bar{\ell}_{\text{PGDM}} = 118$ and $\bar{\ell}_{\text{ADMM}} = 11$ are calculated based on Theorem 3.

Table 3.1: Parameter values for the generalized stopping criterion in the double integrator case study. The validity of the parameters was verified post hoc by confirming that condition (3.16) was satisfied throughout the entire simulation.

Method	**Parameter**	**Value**
General	η_1	0.5
	η_2	0.3
	γ	1.0
PGDM	L	500
	μ	1
	κ_{PGDM}	0.996
	$\bar{\ell}_{\text{PGDM}}$	118
ADMM	ρ	14.37
	κ_{ADMM}	0.958
	$\bar{\ell}_{\text{ADMM}}$	11

Figure 3.3 shows the closed-loop control profiles for the nominal MPC, PGDM($\bar{\ell}$)-based MPC, and ADMM($\bar{\ell}$)-based MPC. To evaluate the accuracy of the suboptimal solutions, PGDM and ADMM were also simulated using a large maximum iteration limit $\ell_{\max} = 15\,000$ (referred to as "unbounded"). These results are not shown in the figure, as the control profiles are visually indistinguishable from the nominal MPC baseline, confirming that the unbounded approaches achieve near-optimal performance.

The fixed-iteration PGDM($\bar{\ell}$) and ADMM($\bar{\ell}$) approaches yield slightly suboptimal control trajectories yet preserve asymptotic stability and successfully drive the system

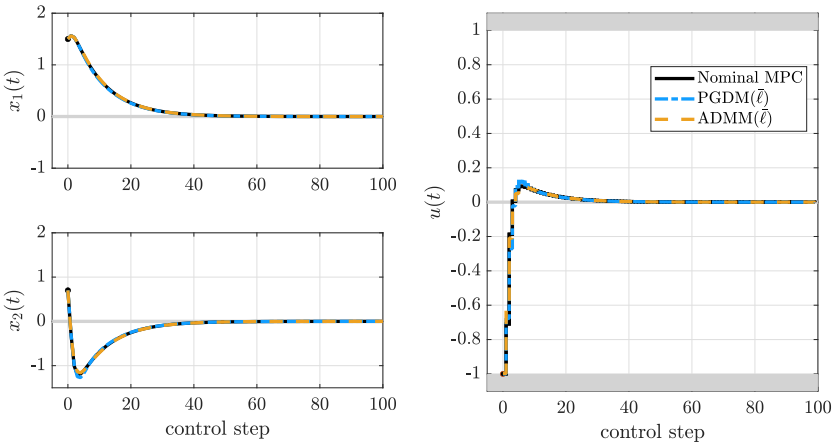


Figure 3.3: Comparison of control performance for the double integrator system. The nominal MPC (reference) is shown as a black solid line. The suboptimal MPC using PGDM with $\bar{\ell}_{\mathrm{PGDM}}$ iterations is depicted by a blue dash-dotted line, while the ADMM-based MPC with $\bar{\ell}_{\mathrm{ADMM}}$ iterations is represented by a yellow dashed line.

states to the origin.

Figure 3.4 illustrates the number of real-time iterations (RTIs) per control step for both the unbounded and $\bar{\ell}$-bounded methods. The unbounded PGDM-based MPC requires a total of 13 560 iterations, while PGDM($\bar{\ell}$) reduces this to 5 370 iterations—a savings of over 60%. Similarly, the ADMM-based MPC requires 3 175 iterations in total, while ADMM($\bar{\ell}$) reduces this to just 947 iterations, yielding a 70% reduction. These results demonstrate that the proposed generalized stopping criterion enables a significant reduction of computational effort while retaining closed-loop stability.

Table 3.2: Summary of iteration and suboptimality metrics for the double integrator case study using the generalized stopping criterion.

Method	Avg. Iter	Max Iter	Total Iter	Saved (%)	σ_{avg} (%)	$\sigma_{\max}$ (%)
ADMM	32.00	106.00	3175.00			
ADMM($\bar{\ell}$)	9.00	11.00	947.00	70.17	0.45	3.15
PGDM	136.00	1315.00	13560.00			
PGDM($\bar{\ell}$)	54.00	118.00	5370.00	60.40	0.65	3.81

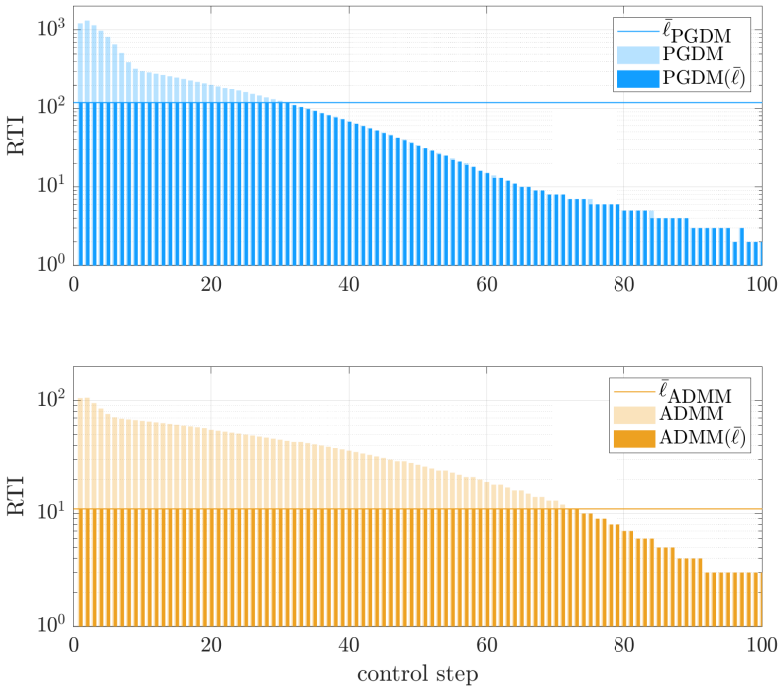


Figure 3.4: Number of real-time iterations (RTI) per control step for the double integrator case study. The figure compares MPC solved using ADMM and PGDM with a tolerance-based stopping criterion to suboptimal MPC using PGDM($\bar{\ell}$) and ADMM($\bar{\ell}$) with a fixed number of iterations.

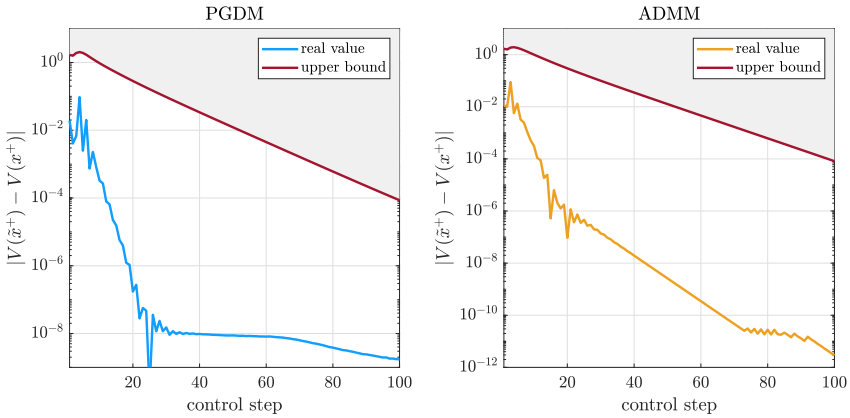


Figure 3.5: Visualization of the deviation $|V(\tilde{x}^+) - V(x^+)|$ at each control step, along with the theoretical upper bound computed using formula (3.16), based on the parameters listed in Table 3.3, for the double integrator system using PGDM with $\bar{\ell}_{\text{PGDM}}$ and ADMM with $\bar{\ell}_{\text{ADMM}}$.

Table 3.2 summarizes the number of iterations and suboptimality rates for both the unbounded and fixed-iteration variants of PGDM and ADMM. The evaluation includes the average and maximum number of iterations, total iterations per simulation, percentage of saved iterations, average suboptimality per control step, and worst-case suboptimality. The results are based on 200 distinct initial conditions $x(t)$, each corresponding to a different segment of the piecewise affine (PWA) control law.

Suboptimality is quantified by the relative value function deviation expressed by the following expression $|V(\tilde{x}^+) - V(x^+)|/|V(x^+)|$. For the unbounded methods, suboptimality is considered negligible ($\leq 10^{-6}$). The $\bar{\ell}$-bounded methods produce low average suboptimality rates, though in some cases, the worst-case suboptimality reaches higher values. If higher solution accuracy is required, the number of fixed iterations can be increased, maintaining asymptotic stability while further reducing suboptimality.

Finally, we emphasize that the generalized stopping criterion provides a provable lower bound on the required number of iterations to ensure closed-loop stability. However, this bound is typically conservative. In practice, a smaller number of iterations may be sufficient to ensure stability, even if it is not theoretically guaranteed.

Figure 3.5 illustrates the real-time value function gap $|V(\tilde{x}^+) - V(x^+)|$ compared to the theoretical upper bound from (3.16), across all control steps. The results confirm

that the actual value function gap remains below the computed upper bound in every instance, validating the stability criterion.

As previously discussed, determining the parameters η_1, η_2, and γ can be challenging for more complex systems. In this case study, they were chosen conservatively and verified by checking that the inequality (3.16) holds throughout the simulation.

3.4.2 Spring-Damper System

The second case study is designed to demonstrate the applicability of the proposed generalized stopping criterion in more complex scenarios. Specifically, we consider a system composed of $\bar{I} = 5$ connected vehicles, each with mass $m_\mathrm{v} = 1\,\mathrm{kg}$, spring constant $k_\mathrm{v} = 3\,\mathrm{N/m}$, damping coefficient $d_\mathrm{v} = 3\,\mathrm{Ns/m}$, and an Euler discretization time step of $T_\mathrm{s} = 0.1\,\mathrm{s}$. The resulting system has $n_\mathrm{x} = 10$ state variables and $n_\mathrm{u} = 5$ control inputs.

The system dynamics is modeled such that only the last vehicle is actuated. Therefore, the first four control inputs remain zero throughout the simulation, as only the fifth input affects the system dynamics. The non-zero blocks of the system matrices are defined as

$$A_{i,i} = \mathbb{I} + T_\mathrm{s} \begin{bmatrix} 0 & 1 \\ -2\frac{k_\mathrm{v}}{m_\mathrm{v}} & -2\frac{d_\mathrm{v}}{m_\mathrm{v}} \end{bmatrix}, \quad B_i = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \tag{3.39}$$

$$A_{\bar{I},\bar{I}} = \mathbb{I} + T_\mathrm{s} \begin{bmatrix} 0 & 1 \\ -\frac{k_\mathrm{v}}{m_\mathrm{v}} & -\frac{d_\mathrm{v}}{m_\mathrm{v}} \end{bmatrix}, \quad B_{\bar{I}} = \begin{bmatrix} 0 \\ \frac{T_\mathrm{s}}{m_\mathrm{v}} \end{bmatrix}, \tag{3.40}$$

$$A_{i-1,i} = A_{i,i+1} = T_\mathrm{s} \begin{bmatrix} 0 & 1 \\ \frac{k_\mathrm{v}}{m_\mathrm{v}} & \frac{d_\mathrm{v}}{m_\mathrm{v}} \end{bmatrix}. \tag{3.41}$$

The input constraint set is defined as $\mathbb{U} = \{u \in \mathbb{R} \mid -1.5 \leq u \leq 0.5\}$, and the MPC cost function is constructed using

$$Q_i = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}, \quad R_i = 1.$$

The terminal penalty matrix P is obtained as the solution to the discrete-time Riccati equation (2.8), and the prediction horizon is set to $N = 40$.

Table 3.3 lists the parameters used for evaluating the generalized stopping criterion in this case study, following the same structure as in the double integrator system.

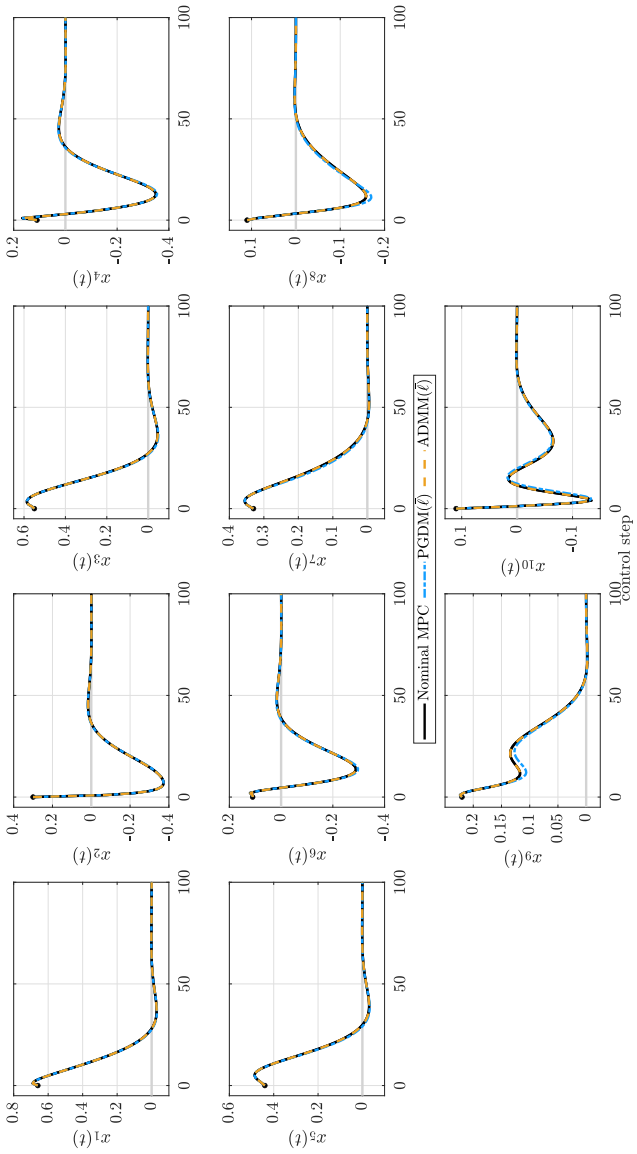


Figure 3.6: Comparison of closed-loop trajectories of the system states for the more complex spring-damper system. The nominal MPC (reference) is shown as a black solid line. The suboptimal MPC using PGDM with $\bar{\ell}_{\text{PGDM}}$ iterations is depicted by a blue dash-dotted line, while the ADMM-based MPC with $\bar{\ell}_{\text{ADMM}}$ iterations is represented by a yellow dashed line.

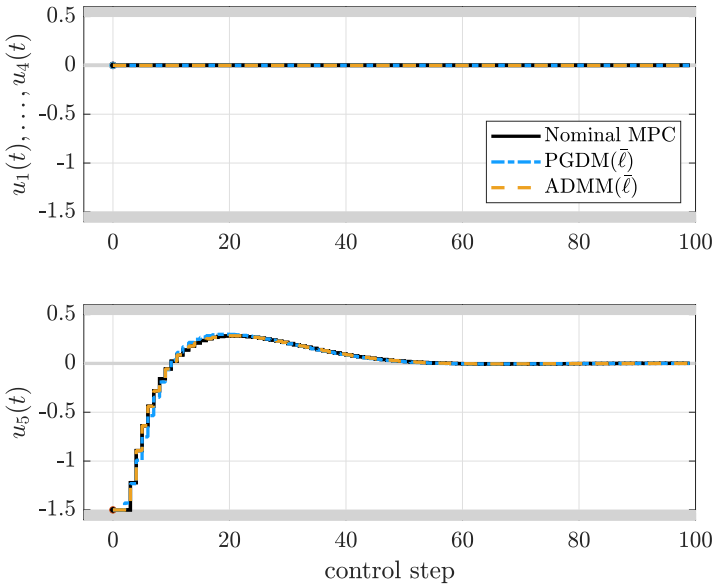


Figure 3.7: Comparison of closed-loop trajectories of the control actions for the more complex spring-damper system. The nominal MPC (reference) is shown as a black solid line. The suboptimal MPC using PGDM with $\bar{\ell}_{\mathrm{PGDM}}$ iterations is depicted by a blue dash-dotted line, while the ADMM-based MPC with $\bar{\ell}_{\mathrm{ADMM}}$ iterations is represented by a yellow dashed line.

Table 3.3: Parameter values for the generalized stopping criterion in the spring-damper case study. The validity of the parameters was verified post hoc by confirming that condition (3.16) was satisfied throughout the entire simulation.

Method	Parameter	Value
General	η_1	8
	η_2	8
	γ	9
PGDM	L	500
	μ	0.4
	κ_{PGDM}	0.998
	$\bar{\ell}_{\text{PGDM}}$	87
ADMM	ρ	0.5
	κ_{ADMM}	0.985
	$\bar{\ell}_{\text{ADMM}}$	10

Figures 3.6 and 3.7 illustrate the control performance in terms of state and control input trajectories, respectively. As mentioned, only the last control input affects the system dynamics, while the others remain inactive throughout the simulation. The $\bar{\ell}$-bounded ADMM and PGDM methods again produce suboptimal trajectories but maintain asymptotic stability and ensure convergence of all state variables to the origin.

Figure 3.8 presents the number of real-time iterations (RTIs) per control step for the unbounded and $\bar{\ell}$-bounded variants of PGDM and ADMM. The total number of iterations for PGDM without bounds is 40 571, compared to 8 190 iterations for PGDM($\bar{\ell}$), resulting in nearly 80% reduction in RTIs. Similarly, ADMM requires 1 522 RTIs in the unbounded case and only 998 RTIs when bounded by $\bar{\ell}$, leading to a 34% reduction. This again confirms the computational advantage of the generalized stopping criterion, particularly in higher-dimensional systems.

Table 3.4 summarizes the average, maximum, and total iteration counts, along with the percentage of saved iterations and the average and worst-case suboptimality for both methods. The ADMM($\bar{\ell}$) method achieves an almost negligible suboptimality (0.04%), while PGDM($\bar{\ell}$) shows significantly higher maximum suboptimality. If such deviations are unacceptable for a given application, one can again increase the number of fixed iterations accordingly, which still ensures asymptotic stability as per Theorem 3.

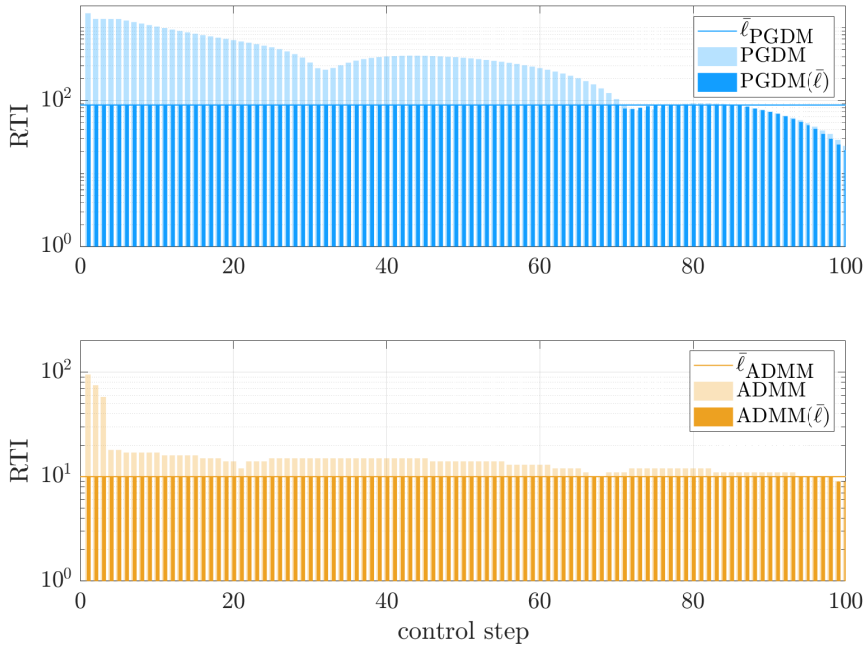


Figure 3.8: Number of real-time iterations (RTI) per control step for the spring-damper case study. The figure compares MPC solved using ADMM and PGDM with a tolerance-based stopping criterion to suboptimal MPC using PGDM($\bar{\ell}$) and ADMM($\bar{\ell}$) with a fixed number of iterations.

Table 3.4: Summary of iteration and suboptimality metrics for the spring-damper case study using the generalized stopping criterion.

Method	Avg. Iter	Max Iter	Total Iter	Saved (%)	σ_{avg} (%)	$\sigma_{\max}$ (%)
ADMM	15.00	95.00	1522.00			
ADMM($\bar{\ell}$)	10.00	10.00	998.00	34.43	0.00	0.04
PGDM	406.00	1573.00	40571.00			
PGDM($\bar{\ell}$)	82.00	87.00	8190.00	79.81	3.48	22.69

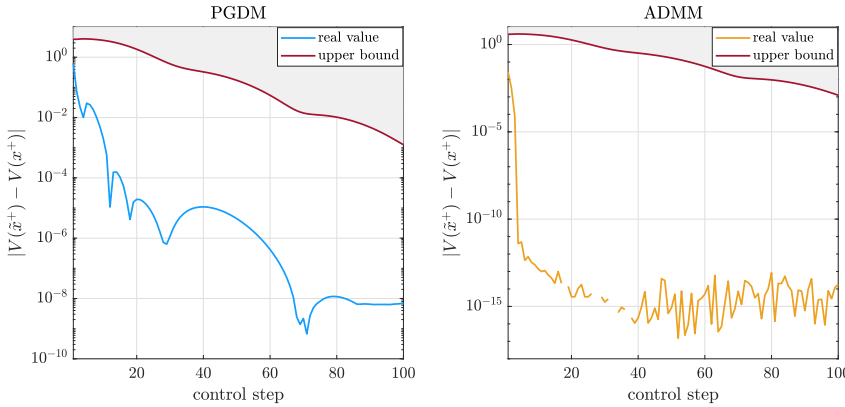


Figure 3.9: Visualization of the deviation $|V(\tilde{x}^+) - V(x^+)|$ at each control step, along with the theoretical upper bound computed using formula (3.16), based on the parameters listed in Table 3.3, for the string-damper system using PGDM with $\bar{\ell}_{\text{PGDM}}$ and ADMM with $\bar{\ell}_{\text{ADMM}}$.

Figure 3.9 visualizes the actual value function gap $|V(\tilde{x}^+) - V(x^+)|$ along with the theoretical upper bound derived in (3.16). It confirms that, for every control step, the deviation remains strictly below the theoretical bound. As with the previous case study, parameters η_1, η_2, and γ were conservatively selected and then verified post hoc to ensure that inequality (3.16) was satisfied throughout the simulation.

All the results highlight the benefits of using the concept of generalized stopping criterion in scenarios with more complex systems.

3.5 Discussion

This chapter has introduced a generalized stopping criterion for the real-time implementation of linear-quadratic MPC under input constraints. The proposed method offers a fixed-iteration stopping criterion suitable to a broad class of linearly convergent first-order optimization algorithms, allowing for the synthesis of suboptimal control policies with guaranteed asymptotic closed-loop stability.

Compared to existing stopping criteria that are algorithm-specific or limited to particular problem formulations, our stopping criterion is general and provably effective. It requires only mild assumptions on the MPC design and system structure. It offers a constructive formula to compute the minimum number of iterations needed to ensure

value function descent and, consequently, stability independently of the initial condition. This contributes toward bridging the gap between theoretical guarantees and the practical, resource-constrained deployment of MPC, especially for embedded or distributed systems where predictable and bounded computational load is essential.

To demonstrate its practicality, we implemented the stopping criterion in two widely used first-order methods, projected gradient descent and the ADMM, and evaluated their closed-loop performance on the benchmark double integrator and spring-damper systems. The numerical results confirm that the number of required iterations can be reduced by over 70% while maintaining stability and introducing suboptimality less than 4%. These results show the effectiveness of the proposed generalized stopping criterion.

Chapter 4

A Real-Time Parallelizable MPC for Embedded Systems

This chapter presents a control framework that prioritizes computational efficiency while maintaining reliable closed-loop performance. The presented control scheme builds upon the concept introduced in Jiang et al. (2021), with crucial extensions that strengthen the theoretical guarantees of stability and feasibility and streamline the control action evaluation process. The proposed framework utilizes explicit MPC theory, where control actions are derived from precomputed piecewise affine control laws rather than solving optimization problems online. As opposed to conventional explicit MPC, however, this method avoids the exponential growth in complexity with respect to the prediction horizon by distributing the control problem along the prediction horizon. As a result, the prediction horizon no longer contributes to the complexity of the control scheme. To further enhance real-time applicability, the method incorporates the generalized stopping criterion introduced in the previous chapter. This enables the early termination of optimization routines used to solve MPC problems, providing formal guarantees of closed-loop stability and feasibility. It is essential to note that the use of explicit control laws ensures optimality under nominal conditions. In contrast, suboptimality arises only due to the fixed-iteration scheme introduced by the generalized stopping criterion. The framework is designed not only as a theoretical contribution but also as a practical solution, supported by an open-source software toolbox called `ParExMPC`. The implementation provides a user interface in `MATLAB` with efficient, low-level routines written in `C` to ensure the applicability of `ParExMPC` on embedded hardware. The results demonstrate that `ParExMPC` can outperform state-of-the-art implicit MPC solvers in terms of runtime, making it a viable candidate for time-critical applications requiring fast and reliable control decisions. The contents of this chapter draw from the findings and results published in Jiang, Fedorová, Su, Oravec, Houska, and Jones (2025b).

4.1 Problem Formualtion

This section presents the mathematical formulation underlying the proposed control strategy for solving the MPC problem defined in (2.7). The strategy builds upon the approach introduced in Jiang et al. (2021), with a key extension: the dualization of the initial condition. This modification enhances the flexibility of the method in real-time settings and presents a mechanism for maintaining feasibility, even in suboptimal scenarios. The considered MPC formulation is defined as follows

$$
\begin{aligned}
V(x(t)) = \min_{x_k, u_k} \quad & x_N^\top P x_N + \sum_{k=0}^{N-1} \left(x_k^\top Q x_k + u_k^\top R u_k \right) && && \text{(4.1a)} \\
\text{s.t.} \quad & x_{k+1} = A x_k + B u_k, \ | \ \lambda_{k+1}, && k \in \{0, \dots, N-1\} && \text{(4.1b)} \\
& x_k \in \mathbb{X}, && k \in \{0, \dots, N-1\} && \text{(4.1c)} \\
& u_k \in \mathbb{U}, && k \in \{0, \dots, N-1\} && \text{(4.1d)} \\
& x_N \in \mathbb{X}_N, && && \text{(4.1e)} \\
& x_0 = x(t), \qquad | \ \lambda_0. && && \text{(4.1f)}
\end{aligned}
$$

A novel aspect of this formulation, compared to the one in (2.7) and one presented in Jiang et al. (2021), is the introduction of the Lagrange multiplier λ_0 associated with the initial condition, which plays a key role in the dual structure of the problem and contributes to the theoretical foundations of the proposed algorithm.

To simplify the notation, the following shorthand representation of the cost function is introduced as

$$
F(x, u) = x_N^\top P x_N + \sum_{k=0}^{N-1} \left(x_k^\top Q x_k + u_k^\top R u_k \right), \tag{4.2}
$$

where $x = [x_0^\top, ..., x_N^\top]^\top$ and $u = [u_0^\top, ..., u_{N-1}^\top]^\top$. The shorthand representation of convex conjugate to $F(x, u)$ has a form of

$$
F^\star(x, u) = \max_{x, u} -F(x, u) - \lambda^\top (Gx + Ku), \tag{4.3}
$$

where matrices G and K capture the dynamic coupling structure as

$$
G = \begin{pmatrix} I & 0 & \cdots & & 0 \\ -A & I & & & \vdots \\ & \ddots & \ddots & \ddots & \\ 0 & \cdots & 0 & -A & I \end{pmatrix}, \quad K = \begin{pmatrix} 0 & 0 & \cdots & & 0 \\ -B & 0 & & & \vdots \\ & \ddots & \ddots & \ddots & \\ 0 & \cdots & 0 & 0 & -B \end{pmatrix}.
$$

The functions F and $F^\star$ are strongly convex implying $F(0) = 0$, $F^\star(0) = 0$. The corresponding primal-dual optimizers $(x^\star, u^\star)$ and $\lambda^\star$ of (4.1) are continuous piecewise

affine (PWA) functions of the initial state $x(t)$, defined over a polytopic partition of the feasible state space $\mathbb{X}$ as presented in Borrelli et al. (2003).

To reduce the online computational burden, we propose to solve the problem (4.1) using the ALADIN optimization algorithm. Therefore, we reformulate the optimization problem (4.1) using the augmented Lagrangian function, allowing straightforward distribution as presented in Section 2.2.2 in the following shorthand form

$$\begin{aligned} \min_{x,u} \;& F(x,u)+\lambda^\top Gx+\lambda^\top Ku+\|x-\bar{x}\|_{\mathcal{Q}}^2+\|u-\bar{u}\|_{\mathcal{R}}^2 \\ \text{s.t.} \;& \begin{cases} x \in \mathbb{X}\cdots\mathbb{X}\times\mathbb{X}_N, \\ u \in \mathbb{U}\times\cdots\times\mathbb{U}, \end{cases} \end{aligned} \tag{4.4}$$

where $(\bar{x},\bar{u})$ denotes reference vectors, which serve as global coordination variables and $\mathcal{Q}=\nabla_x^2 F(x,u)$, $\mathcal{R}=\nabla_u^2 F(x,u)$ are Hessians corresponding to state and input variables, respectively. This formulation (4.4) is essential for theoretical analysis in further sections.

To make the separable structure of the MPC problem (4.1) explicit, we now present a mathematically equivalent, detailed formulation using augmented Lagrangian function as follows

$$\begin{aligned} \min_{x,u} \;& x_N^\top P x_N+\sum_{k=0}^{N-1}\left(x_k^\top Q x_k+u_k^\top R u_k\right)+\sum_{k=0}^{N-1}\left(\lambda_k^\top x_k-\lambda_{k+1}^\top(Ax_k+Bu_k)\right) \\ &+\lambda_N^\top x_N+\sum_{k=0}^{N-1}\left(\|x_k-\bar{x}_k\|_Q^2+\|u_k-\bar{u}_k\|_R^2\right)+\|x_N-\bar{x}_N\|_P^2 \\ \text{s.t.} \;& x_k\in\mathbb{X},\; u_k\in\mathbb{U},\; x_N\in\mathbb{X}_N, \quad k\in\{0,\dots,N-1\}. \end{aligned} \tag{4.5}$$

Consequently, the resulting subproblems share the same structure for $k\in\{0,\dots,N-1\}$, while the terminal step at $k=N$ differs. We, therefore, formulate subproblems that fit the ALADIN framework as follows

$$\min_{x_k^\ell\in\mathbb{X},u_k^\ell\in\mathbb{U}} \left\|\begin{bmatrix}x_k^\ell\\u_k^\ell\end{bmatrix}\right\|_{\mathbf{\Sigma}}^2+\begin{bmatrix}\lambda_k^\ell\\\lambda_{k+1}^\ell\end{bmatrix}^\top\begin{bmatrix}I & 0\\-A & -B\end{bmatrix}\begin{bmatrix}x_k^\ell\\u_k^\ell\end{bmatrix}+\left\|\begin{bmatrix}x_k^\ell-\bar{x}_k^\ell\\u_k^\ell-\bar{u}_k^\ell\end{bmatrix}\right\|_{\mathbf{\Sigma}}^2, \tag{4.6a}$$

$$\min_{x_N^\ell\in\mathbb{X}_N} \left\|x_N^\ell\right\|_P^2+{\lambda_N^\ell}^\top x_N^\ell+\left\|x_N^\ell-\bar{x}_N^\ell\right\|_P^2, \tag{4.6b}$$

with the penalty matrix $\mathbf{\Sigma}=\operatorname{diag}(Q,R)$. The resulting subproblems can be solved efficiently and independently across the prediction horizon due to their decoupled structure. In addition, strong convexity is preserved for all subproblems. Throughout

this chapter, variables are indexed using superscripts to denote the ALADIN iteration ℓ and subscripts to indicate the corresponding k-th step of the prediction horizon, as in x_k^ℓ. To reduce the computational effort required for solving (4.6) during the online phase, the corresponding QPs are pre-solved offline using multi-parametric programming. This approach yields explicit solution maps of the form

$$(x_k^\star, u_k^\star)(\theta_k) = \underset{x_k \in \mathbb{X}, u_k \in \mathbb{U}}{\arg\min} \; F_k(x_k, u_k) + \theta_{k,1}^\top x_k + \theta_{k,2}^\top u_k + \|x_k - \theta_{k,3}\|_Q^2 + \|u_k - \theta_{k,4}\|_R^2, \tag{4.7a}$$

$$x_N^\star(\theta_N) = \underset{x_N \in \mathbb{X}_N}{\arg\min} \; F_k(x_N) + \theta_{N,1}^\top x_N + \|x_N - \theta_{N,2}\|_P^2, \tag{4.7b}$$

where parameters $\theta_k = [\theta_{k,1}^\top, \theta_{k,2}^\top, \theta_{k,3}^\top, \theta_{k,4}^\top]^\top$ and $\theta_N = [\theta_{N,1}^\top, \theta_{N,2}^\top]^\top$ contain relevant information for each stage k, including system dynamics, Lagrange multipliers, and reference vectors. Since the structure of the decoupled optimization problems remains identical across all stages $k \in \{0, \dots, N-1\}$, a single explicit solution map $(x_1^\star, u_1^\star)$ is sufficient, and it can be reused for all corresponding subproblems. In contrast, the final-stage problem at $k = N$ has a distinct formulation and requires a separate parametric solution $x_N^\star$.

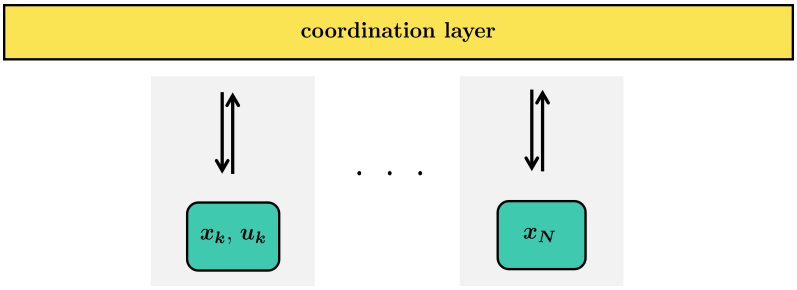


Figure 4.1: Visualization of the time-domain decomposition of the MPC problem with dualization of the initial condition. In contrast to Fig. 2.1, this formulation involves only two types of subproblem structures.

This separation introduces a computational advantage: the explicit maps only need to be generated for two representative QPs as visualized in Fig. 4.1 due to the dualization of the initial condition. It allows for a more compact problem formulation while preserving its structure. As a result, both the offline computation time and the online memory requirements are reduced compared to the method proposed in Jiang et al. (2021), where explicit maps are constructed for three different QPs (following structure illustrated in Fig 2.1). Given that strong convexity is preserved, the solution maps $(x_1^\star, u_1^\star)$ and $x_N^\star$ are piecewise affine functions defined over a finite number of polytopic regions, commonly referred to as critical regions. These maps are computed offline,

and the appropriate control input is evaluated during the online phase by solving a point location problem that identifies the active region corresponding to the current parameter vector as presented in Borrelli (2003).

Once the local QPs have been formulated and solved, their solutions must be coordinated to ensure consistency across the prediction horizon. This coordination is performed as described in Step 3) of Algorithm 2, which, for the MPC problem (4.1), leads to the following coupled equality-constrained QP

$$\min_{\bar{x}^{\ell+1},\bar{u}^{\ell+1}} \quad \left\|\bar{x}_N^{\ell+1} - 2x_N^{\ell} + \bar{x}_N^{\ell}\right\|_P^2 + \sum_{k=0}^{N-1} \left\| \begin{bmatrix} \bar{x}_k^{\ell+1} - 2x_k^{\ell} + \bar{x}_k^{\ell} \\ \bar{u}_k^{\ell+1} - 2u_k^{\ell} + \bar{u}_k^{\ell} \end{bmatrix} \right\|_{\mathbf{\Sigma}}^2 \tag{4.8a}$$

$$\text{s.t.} \quad \begin{cases} \bar{x}_{k+1}^{\ell+1} = A\bar{x}_k^{\ell+1} + B\bar{u}_k^{\ell+1} & | \ \Delta\lambda_{k+1}^{\ell}, \quad k \in \{0,\dots,N-1\}, \\ \bar{x}_0^{\ell+1} = x(t) & | \ \Delta\lambda_0^{\ell}. \end{cases} \tag{4.8b}$$

The resulting coupled equality-constrained QP preserves strong convexity and can be effectively solved using standard numerical techniques. The specific approach adopted in our implementation is detailed in Section 4.4.

4.2 Algorithm Overview

The complete procedure of the proposed control strategy is summarized in Algorithm 4, which integrates an augmented Lagrangian-based optimization framework with precomputed explicit solution maps for efficient real-time execution.

The algorithm begins with the initialization of primal and dual variables as

$$x^1 = [x_0^1,\dots,x_N^1], \quad u^1 = [u_0^1,\dots,u_{N-1}^1], \quad \lambda^1 = [\lambda_0^1,\dots,\lambda_N^1].$$

These variables provide the initial guess for the optimization problem at the beginning of each control step. Then, the rescaling procedure is performed in Step 1) to ensure numerical robustness and facilitate convergence as explained in Jiang et al. (2021). This step adjusts the magnitude of the initial guess based on the current system state and a tuning parameter $\gamma > 0$ only if $f^1 = F(x^1,u^1) + F^\star(\lambda^1) \geq \gamma^2\|x(t)\|_Q^2$, as follows

$$\bar{x}^1 = \bar{x}^1\sqrt{\left(\gamma^2\|x(t)\|_Q^2\right)/f^1}, \tag{4.14}$$

$$\bar{u}^1 = \bar{u}^1\sqrt{\left(\gamma^2\|x(t)\|_Q^2\right)/f^1}, \tag{4.15}$$

$$\bar{\lambda}^1 = \bar{\lambda}^1\sqrt{\left(\gamma^2\|x(t)\|_Q^2\right)/f^1}. \tag{4.16}$$

Algorithm 4 Parallel Real-Time Explicit Model Predictive Control.

Offline:

- Set initial guesses $\bar{x}^1 = [\bar{x}_0^1, \ldots, \bar{x}_N^1]$, $\bar{u}^1 = [\bar{u}_0^1, \ldots, \bar{u}_{N-1}^1]$ and $\lambda^1 = [\lambda_0^1, \ldots, \lambda_N^1]$, a constant $\gamma > 0$, penalty matrix $\mathbf{\Sigma} = \mathrm{diag}(Q, R)$ and generate the explicit solution maps $(x^\star, u^\star)(\theta_k)$, $x_N^\star(\theta_N)$ using (4.7a) and (4.7b) respectively.

Online:

1) Get the state measurement $x(t)$ at the current time t, set $f^1 = F(x^1, u^1) + F^\star(\lambda^1)$. If $f^1 \geq \gamma^2 \|x(t)\|_Q^2$ set

$$\bar{x}^1 = \bar{x}^1 \sqrt{\left(\gamma^2 \|x(t)\|_Q^2\right) \big/ f^1}, \tag{4.9}$$

$$\bar{u}^1 = \bar{u}^1 \sqrt{\left(\gamma^2 \|x(t)\|_Q^2\right) \big/ f^1}, \tag{4.10}$$

$$\bar{\lambda}^1 = \bar{\lambda}^1 \sqrt{\left(\gamma^2 \|x(t)\|_Q^2\right) \big/ f^1}. \tag{4.11}$$

2) **For** $\ell = 1 \to \bar{\ell}$

(a) Compute the parameters for all $k \in \{0, \ldots, N-1\}$

$$\theta_{k,1}^\ell = \lambda_k^\ell - A^\top \lambda_{k+1}^\ell,\ \theta_{k,2}^\ell = -B^\top \lambda_{k+1}^\ell,\ \theta_{k,3}^\ell = \bar{x}_k^\ell,\ \theta_{k,4}^\ell = \bar{u}_k^\ell, \tag{4.12a}$$

$$\theta_{N,1}^\ell = \lambda_N^\ell,\ \theta_{N,2}^\ell = \bar{x}_N^\ell, \tag{4.12b}$$

and evaluate the vectors of optimizers

$$(x_k^\ell, u_k^\ell) = (x_1^\star, u_1^\star)(\theta_k^\ell),\ \ x_N^\ell = x_N^\star(\theta_N^\ell). \tag{4.13}$$

(b) Solve the coupled LQR problem (4.8) to provide the solution for $\bar{x}^{\ell+1}$, $\bar{u}^{\ell+1}$, $\lambda^{\ell+1}$ and set $\Delta\lambda^\ell = \lambda^\ell + \Delta\lambda^\ell$.

End

3) Apply $u(t) = u_0^{\bar{\ell}}$ to the real process.

4) Set $\bar{x}^1 = [x_1^{\bar{\ell}}, \ldots, x_N^{\bar{\ell}}, 0]$, $\bar{u}^1 = [u_1^{\bar{\ell}}, \ldots, u_{N-1}^{\bar{\ell}}, 0]$ and $\lambda^1 = [\lambda_1^{\bar{\ell}}, \ldots, \lambda_N^{\bar{\ell}}, 0]$ and go to Step 1.

The existence of a suitable γ guaranteeing that the rescaling remains within relevant bounds is formalized by the following assumption.

Assumption 6. *The constant γ in Algorithm 4 satisfies*

$$F(x^\star(x(t)), u^\star(x(t))) + F^\star(\lambda^\star(x(t))) \leq \gamma^2 x(t)^\top Q x(t). \tag{4.17}$$

This assumption is conceptually equivalent to Assumption 4 introduced in Chapter 3, but it is specifically tailored to the second-order ALADIN method. As previously discussed, such a constant γ always exists and can be precomputed offline.

After rescaling, the algorithm proceeds by iteratively solving the MPC problem using the ALADIN method. As described in Section 2.2.2, ALADIN is executed through two primary phases. In Step 2a) of Algorithm 4, decoupled local subproblems are evaluated using precomputed explicit solution maps, which drastically reduce the required online computational effort. In Step 2b), a coupled QP is solved to coordinate the local updates and enforce consistency across the prediction horizon. Both primal and dual variables are updated accordingly during this step.

After performing a fixed number of iterations $\bar{\ell}$, the evaluated control input $u_0^{\bar{\ell}}$ is applied to the system in Step 3). The algorithm then proceeds to Step 4), where a warm-starting procedure is employed for the next iteration. Compared to classical warm-starting strategies such as that in Allan et al. (2017), this approach shifts the current solution forward in time and adds zeros in the time shift as follows

$$\bar{x}^1 = [x_1^{\bar{\ell}}, \ldots, x_N^{\bar{\ell}}, 0], \quad \bar{u}^1 = [u_1^{\bar{\ell}}, \ldots, u_{N-1}^{\bar{\ell}}, 0], \quad \lambda^1 = [\lambda_1^{\bar{\ell}}, \ldots, \lambda_N^{\bar{\ell}}, 0].$$

This warm-start strategy is similar to that proposed in Braun et al. (2018) for ADMM-based distributed MPC, and it exploits the continuity of state and input trajectories of the MPC problem to enhance convergence behavior in the next iteration.

The theoretical properties of Algorithm 4, including feasibility and closed-loop stability, are investigated in the following section.

4.3 Theoretical Analysis

Before proceeding with the theoretical analysis of Algorithm 4, we introduce compact notation for the stage cost functions. For each step $k \in= \{0, \ldots, N-1\}$, the stage cost is defined as

$$F_k(x_k, u_k) = x_k^\top Q x_k + u_k^\top R u_k, \tag{4.18}$$

while the terminal cost is given by

$$F_N(x_N) = x_N^\top P x_N. \tag{4.19}$$

To establish the connection between the decoupled and coupled optimization variables, we present the stationarity conditions for the problem (4.8) in Step 2b) of Algorithm 4 as

$$\mathcal{Q}(\bar{x}^{\ell+1} - 2x^\ell + \bar{x}^\ell) + G^\top \Delta\lambda^\ell = 0, \tag{4.20a}$$
$$\mathcal{R}(\bar{u}^{\ell+1} - 2u^\ell + \bar{u}^\ell) + K^\top \Delta\lambda^\ell = 0, \tag{4.20b}$$

where $\Delta\lambda^\ell = \lambda^{\ell+1} - \lambda^\ell$ denotes the dual variable update.

Given that $\mathcal{Q}$ and $\mathcal{R}$ are both symmetric and positive definite, we yield the solutions x^ℓ and u^ℓ in the following form

$$x^\ell = \tfrac{1}{2}\mathcal{Q}^{-1}G^\top(\lambda^{\ell+1} - \lambda^\ell) + \tfrac{1}{2}(\bar{x}^{\ell+1} + \bar{x}^\ell), \tag{4.21a}$$
$$u^\ell = \tfrac{1}{2}\mathcal{R}^{-1}K^\top(\lambda^{\ell+1} - \lambda^\ell) + \tfrac{1}{2}(\bar{u}^{\ell+1} + \bar{u}^\ell). \tag{4.21b}$$

These expressions will later be used to support the convergence and stability analysis of the Algorithm 4.

4.3.1 Convergence Properties

This section analyses the convergence properties of Step 2) in Algorithm 4, which is used to solve the online MPC problem defined in (4.1). We begin by recalling the following key result, which formally establishes a contraction property for the iterates of the proposed algorithm.

Theorem 4 (Theorem 1 in Jiang et al. (2021))**.** *Let the problem* (4.1) *be feasible with the unique primal* $x^\star$*,* $u^\star$*, and dual* $\lambda^\star$ *solutions. Then, there exists a constant* $\kappa \in (0, 1)$ *such that evaluating the Step 2) of Algorithm 4 yields the following inequality*

$$\begin{aligned} &F\left(x^{\ell+1} - x^\star,\, u^{\ell+1} - u^\star\right) + F^\star\left(\lambda^{\ell+1} - \lambda^\star\right) \\ &\quad \leq \kappa \cdot \left(F\left(x^\ell - x^\star,\, u^\ell - u^\star\right) + F^\star\left(\lambda^\ell - \lambda^\star\right)\right). \end{aligned} \tag{4.22}$$

that holds for all $\ell \geq 2$.

Proof. To prove Theorem 4, we begin by introducing auxiliary functions that reflect the decoupled optimization problems evaluated at iteration ℓ. Specifically, for each step $k \in \{0, \dots, N-1\}$, we define

$$\begin{aligned}\mathcal{F}_k(\phi_k^{\mathrm{x}}, \phi_k^{\mathrm{u}}) = F_k(\phi_k^{\mathrm{x}}, \phi_k^{\mathrm{u}}) + (\lambda_k^\ell - A^\top \lambda_{k+1}^\ell)^\top \phi_k^{\mathrm{x}} - (B^\top \lambda_{k+1}^\ell)^\top \phi_k^{\mathrm{u}} \\ + \nabla_{\mathrm{x}} F_k(x_k^\ell - \bar{x}_k^\ell, u_k^\ell - \bar{u}_k^\ell)^\top \phi_k^{\mathrm{x}} + \nabla_{\mathrm{u}} F_k(x_k^\ell - \bar{x}_k^\ell, u_k^\ell - \bar{u}_k^\ell)^\top \phi_k^{\mathrm{u}},\end{aligned}$$

and for the terminal stage $k = N$,

$$\mathcal{F}_N(\phi_N^{\mathrm{x}}) = F_N(\phi_N^{\mathrm{x}}) + (\lambda_N^\ell)^\top \phi_N^{\mathrm{x}} + \nabla F_N(x_N^\ell - \bar{x}_N^\ell)^\top \phi_N^{\mathrm{x}}.$$

Since x_k^ℓ and u_k^ℓ are the minimizers of the local QPs in Step 2a) of Algorithm 4, they also minimize the corresponding auxiliary function $\mathcal{F}_k$ over their feasible sets $\mathbb{X}$ and $\mathbb{U}$. Since these functions are strongly convex, the following inequality must hold

$$\begin{aligned}\sum_{k=0}^{N-1} \mathcal{F}_k(x_k^\ell, u_k^\ell) + \mathcal{F}_N(x_N^\ell) + \sum_{k=0}^{N-1} F_k(x_k^\ell - x_k^\star, u_k^\ell - u_k^\star) + F_N(x_N^\ell - x_N^\star) \\ \leq \sum_{k=0}^{N-1} \mathcal{F}_k(x_k^\star, u_k^\star) + \mathcal{F}_N(x_N^\star). \qquad (4.23)\end{aligned}$$

To relate the optimal and suboptimal primal-dual pairs, we exploit duality in (4.1). Accordingly, we obtain

$$\begin{aligned}&\sum_{k=0}^{N-1} F_k(x_k^\star, u_k^\star) + F_N(x_N^\star) + \sum_{k=0}^{N-1} \left((\lambda_k^\star - A^\top \lambda_{k+1}^\star)^\top x_k^\star - (B^\top \lambda_{k+1}^\star)^\top u_k^\star\right) \\ &\quad + (\lambda_N^\star)^\top x_N^\star + \sum_{k=0}^{N-1} F_k(x_k^\ell - x_k^\star, u_k^\ell - u_k^\star) + F_N(x_N^\ell - x_N^\star) \\ &\leq \sum_{k=0}^{N-1} F_k(x_k^\ell, u_k^\ell) + F_N(x_N^\ell) \\ &\quad + \sum_{k=0}^{N-1} \left((\lambda_k^\star - A^\top \lambda_{k+1}^\star)^\top x_k^\ell - (B^\top \lambda_{k+1}^\star)^\top u_k^\ell\right) + (\lambda_N^\star)^\top x_N^\ell. \qquad (4.24)\end{aligned}$$

Then, we add both inequalities together and eliminate the identical terms to obtain

$$\begin{aligned}0 \geq & \sum_{k=0}^{N-1} \left((\lambda_k^\ell - \lambda_k^\star - A^\top(\lambda_{k+1}^\ell - \lambda_{k+1}^\star))^\top (x_k^\ell - x_k^\star)\right) \\ & - \sum_{k=0}^{N-1} (B^\top(\lambda_{k+1}^\ell - \lambda_{k+1}^\star))^\top (u_k^\ell - u_k^\star) + \sum_{k=0}^{N-1} \nabla_{\mathrm{x}} F_k(x_k^\ell - \bar{x}_k^\ell, u_k^\ell - \bar{u}_k^\ell)^\top (x_k^\ell - x_k^\star) \\ & + \sum_{k=0}^{N-1} \nabla_{\mathrm{u}} F_k(x_k^\ell - \bar{x}_k^\ell, u_k^\ell - \bar{u}_k^\ell)^\top (u_k^\ell - u_k^\star) \\ & + (\lambda_N^\ell - \lambda_N^\star)^\top (x_N^\ell - x_N^\star) + \nabla F_N(x_N^\ell - \bar{x}_N^\ell)^\top (x_N^\ell - x_N^\star) \\ & + 2 \sum_{k=0}^{N-1} F_k(x_k^\ell - x_k^\star, u_k^\ell - u_k^\star) + 2F_N(x_N^\ell - x_N^\star).\end{aligned} \tag{4.25}$$

Moreover, we substitute the expression for x^ℓ and u^ℓ from the stationarity condition of the coupled QP (4.21) and simplify the previous inequality by using matrix notation, which results in

$$\begin{aligned}-2F(x^\ell - x^\star, u^\ell - u^\star) \geq & (\lambda^\ell - \lambda^\star)^\top G(x^\ell - x^\star) + (\lambda^\ell - \lambda^\star)^\top K(u^\ell - u^\star) \\ & + (x^\ell - x^\star)^\top \mathcal{Q}(x^\ell - x^\star) + (u^\ell - u^\star)^\top \mathcal{R}(u^\ell - u^\star) \qquad (4.26a) \\ \geq & \frac{1}{4}(\lambda^{\ell+1} - \lambda^\ell)^\top (G\mathcal{Q}^{-1}G^\top + K\mathcal{R}^{-1}K^\top)(\lambda^{\ell+1} - \lambda^\ell) \\ & + \frac{1}{4}(\bar{x}^{\ell+1} - \bar{x}^\ell)^\top Q(\bar{x}^{\ell+1} + \bar{x}^\ell - 2x^\star) \\ & + \frac{1}{4}(\bar{u}^{\ell+1} - \bar{u}^\ell)^\top R(\bar{u}^{\ell+1} + \bar{u}^\ell - 2u^\star) \\ & + \frac{1}{2}(\lambda^\ell - \lambda^\star)^\top (G\mathcal{Q}^{-1}G^\top + K\mathcal{R}^{-1}K^\top)(\lambda^{\ell+1} - \lambda^\ell) \qquad (4.26b) \\ = & \frac{1}{2}\left(F(\bar{x}^{\ell+1} - x^\star, \bar{u}^{\ell+1} - u^\star) - F(\bar{x}^\ell - x^\star, \bar{u}^\ell - u^\star)\right) \\ & + \frac{1}{2}\left(F^\star(\lambda^{\ell+1} - \lambda^\star) - F^\star(\lambda^\ell - \lambda^\star)\right). \qquad (4.26c)\end{aligned}$$

Summing over iterations from $\ell = \hat{\ell}$ to $\bar{\ell}$ with $\hat{\ell} \geq 2$ and applying telescoping yields

$$\begin{aligned}-2\sum_{\ell=\hat{\ell}}^{\bar{\ell}} F(x^\ell - x^\star, u^\ell - u^\star) \geq & -\frac{1}{2}\left(F(\bar{x}^{\hat{\ell}} - x^\star, \bar{u}^{\hat{\ell}} - u^\star) + F^\star(\lambda^{\hat{\ell}} - \lambda^\star)\right) \\ & + \frac{1}{2}\left(F(\bar{x}^{\bar{\ell}+1} - x^\star, \bar{u}^{\bar{\ell}+1} - u^\star) + F^\star(\lambda^{\bar{\ell}+1} - \lambda^\star)\right).\end{aligned} \tag{4.27}$$

Finally, since

$$\frac{1}{2}\left(F(\bar{x}^{\bar{\ell}+1} - x^\star, \bar{u}^{\bar{\ell}+1} - u^\star) + F^\star(\lambda^{\bar{\ell}+1} - \lambda^\star)\right) \geq 0, \tag{4.28}$$

we can derive a conservative upper bound without this term as

$$\sum_{\ell=\hat{\ell}}^{\bar{\ell}} F(x^\ell - x^\star, u^\ell - u^\star) \leq \frac{1}{4}\left(F(x^{\hat{\ell}} - x^\star, u^{\hat{\ell}} - u^\star) + F^\star(\lambda^{\hat{\ell}} - \lambda^\star)\right). \tag{4.29}$$

This result verifies that the sequence of primal-dual residuals of Algorithm 4 converges, thereby ensuring that the convergence properties of the ALADIN algorithm are retained under the modified formulation. The remainder of the proof follows the argumentation presented in Jiang et al. (2021), demonstrating that the linear convergence rate established therein continues to hold even with the introduction of initial condition dualization. □

This result implies that the sequence of primal-dual iterates contracts at a linear rate in terms of the primal and dual residuals. As a consequence, if the number of ALADIN iterations was allowed to increase indefinitely, i.e., $\ell \to \infty$, the algorithm would converge to the optimal solution of (4.1) as

$$\lim_{\ell\to\infty} x^\ell = x^\star, \quad \lim_{\ell\to\infty} u^\ell = u^\star, \quad \lim_{\ell\to\infty} \lambda^\ell = \lambda^\star.$$

Furthermore, the constant κ in Theorem 4 is not only guaranteed to exist but can also be explicitly computed using the structural properties of the MPC problem and the ALADIN algorithm. In the following, this convergence property will be utilized to derive additional results on the closed-loop performance and stability of the proposed MPC scheme.

4.3.2 Closed-Loop Performance

Before analyzing the closed-loop behavior of the proposed controller, we introduce a standard assumption regarding the terminal ingredients of the MPC formulation.

Assumption 7. *There exists a feedback control law $\mu : \mathbb{X}_N \to \mathbb{U}$ such that:*

a) The terminal set $\mathbb{X}_N \subseteq \mathbb{X}$ is a control invariant $\mu(\cdot)$, i.e.,

$$x^+ = Ax + B\mu(x) \in \mathbb{X}_N \quad \textit{for all } x \in \mathbb{X}_N,$$

b) The terminal cost $\|\cdot\|_P^2$ *serves as a local Lyapunov function in* $\mathbb{X}_N$*, satisfying*

$$\|Ax + B\mu(x)\|_P^2 \leq \|x\|_P^2 - \left(\|x\|_Q^2 + \|\mu(x)\|_R^2\right), \quad \forall x \in \mathbb{X}_N.$$

Under Assumption 7, applying the optimal control input $u_0^\star$ from the exact solution of the MPC problem (4.1) ensures asymptotic stability through the classic receding horizon strategy (Rawlings et al., 2009). However, in Algorithm 4, we adopt a real-time approach and instead apply the control input $u_0^{\bar{\ell}}$, obtained after a finite number of ALADIN iterations.

The resulting successor state generally differs from the optimal state and is defined in the same way as in Chapter 3 by

$$x^+ \neq \tilde{x}^+ = Ax(t) + Bu_0^{\bar{\ell}}.$$

The following theorem establishes a descent property of the value function under the proposed real-time scheme. Compared to the general formulation in Theorem 3, this result is specifically derived for the second-order ALADIN scheme presented in Algorithm 4.

Theorem 5. *Let Assumptions 1, 3, 5, 6, and 7 hold. Let the MPC problem* (4.1) *be feasible for both the initial state* $x_0 = x(t)$ *and the successor state* $\tilde{x}^+ = Ax(t) + Bu_0^{\bar{\ell}}$*, the constant* $\sigma > 0$ *be such that* $B^\top Q B \preceq \sigma R$*, and constants* $\eta_1, \eta_2 > 0$ *satisfy*

$$|V(\tilde{x}^+) - V(x^+)| \leq \eta_1 \|\tilde{x}^+ - x^+\|_Q + \frac{\eta_2}{2}\|\tilde{x}^+ - x^+\|_Q^2. \tag{4.30}$$

If the constant $\bar{\ell} \in \mathbb{N}$ *satisfies*

$$\bar{\ell} > \frac{2\log\left(2\eta_1\gamma\sqrt{\frac{\sigma(1+\kappa)}{\kappa}} + 2\eta_2\sigma\gamma^2\frac{1+\kappa}{\kappa}\right)}{\log(1/\kappa)}, \tag{4.31}$$

then we have

$$V(\tilde{x}^+) \leq V(x_0) - \beta\|x_0\|_Q^2, \tag{4.32}$$

that asymptotically stabilizes the closed-loop system.

This result demonstrates that, despite the use of approximate control inputs due to the finite number of ALADIN iterations, the closed-loop system retains a stabilizing property in the form of value function descent, provided that $\bar{\ell}$ is selected according to (4.31). Furthermore, the proof structure largely follows the steps outlined in Chapter 3, with modifications to reflect the specific properties of the ALADIN algorithm.

Proof. We begin by expressing the deviation between the suboptimal and optimal successor states as

$$\tilde{x}^+ - x^+ = Ax(t) + Bu_0^\ell - (Ax(t) + Bu_0^\star) = B\left(u_0^\ell - u_0^\star\right) = \tilde{B}(u^\ell - u^\star), \tag{4.33}$$

where $\tilde{B} = [B, 0, \ldots, 0] \in \mathbb{R}^{n_\mathrm{x} \times N n_\mathrm{u}}$ has appropriate dimensions.

Next, we recall the stationary condition from the coupled QP solution in Step 2b) as

$$u^\ell = \frac{1}{2} R^{-1} K^\top (\lambda^{\ell+1} - \lambda^\ell) + \frac{1}{2}(\bar{u}^{\ell+1} + \bar{u}^\ell). \tag{4.34}$$

Combining this with the optimal control input $u^\star$ and introducing the zero term $\lambda^\star - \lambda^\star$, we obtain

$$\begin{aligned}
\tilde{x}^+ - x^+ &= \tilde{B}\left(\frac{1}{2} R^{-1} K^\top (\lambda^{\ell+1} - \lambda^\ell + \lambda^\star - \lambda^\star) + \frac{1}{2}(\bar{u}^{\ell+1} + \bar{u}^\ell - 2u^\star)\right) \\
&= \frac{1}{2}\tilde{B} R^{-1} K^\top (\lambda^{\ell+1} - \lambda^\star) + \frac{1}{2}\tilde{B} R^{-1} K^\top (\lambda^\star - \lambda^\ell) \\
&\quad + \frac{1}{2}\tilde{B}(u^{\ell+1} - u^\star) + \frac{1}{2}\tilde{B}(u^\ell - u^\star).
\end{aligned} \tag{4.35}$$

We now evaluate the deviation between the optimal and suboptimal state trajectories in terms of the Q-weighted norm

$$\begin{aligned}
\|\tilde{x}^+ - x^+\|_Q^2 = \frac{1}{4}&(\lambda^{\ell+1} - \lambda^\star)^\top K R^{-1} \tilde{B}^\top Q \tilde{B} R^{-1} K^\top (\lambda^{\ell+1} - \lambda^\star) \\
&+ \frac{1}{4}(u^{\ell+1} - u^\star)^\top \tilde{B}^\top Q \tilde{B}(u^{\ell+1} - u^\star) \\
&+ \frac{1}{4}(\lambda^\ell - \lambda^\star)^\top K R^{-1} \tilde{B}^\top Q \tilde{B} R^{-1} K^\top (\lambda^\ell - \lambda^\star) \\
&+ \frac{1}{4}(u^\ell - u^\star)^\top \tilde{B}^\top Q \tilde{B}(u^\ell - u^\star).
\end{aligned} \tag{4.36}$$

By moving constant $1/4$ to the left-hand side of equation (4.36), applying the semi-definite bound $B^\top Q B \preceq \sigma R$ and using the structure of $\tilde{B}$, we get

$$\begin{aligned}
4\|\tilde{x}^+ - x^+\|_Q^2 &\leq \sigma(\lambda^{\ell+1} - \lambda^\star)^\top K R^{-1} K^\top (\lambda^{\ell+1} - \lambda^\star) \\
&\quad + \sigma(u^{\ell+1} - u^\star)^\top R(u^{\ell+1} - u^\star) \\
&\quad + \sigma(\lambda^\star - \lambda^\ell)^\top K R^{-1} K^\top (\lambda^\star - \lambda^\ell) \\
&\quad + \sigma(u^\ell - u^\star)^\top R(u^\ell - u^\star) \qquad (4.37a) \\
&\leq \sigma\left(F(x^{\ell+1} - x^\star, u^{\ell+1} - u^\star) + F^\star(\lambda^{\ell+1} - \lambda^\star)\right) \\
&\quad + \sigma\left(F(x^\ell - x^\star, u^\ell - u^\star) + F^\star(\lambda^\ell - \lambda^\star)\right). \qquad (4.37b)
\end{aligned}$$

Using the linear convergence result from Theorem 4, the right-hand side can be upper bounded by

$$4\|\tilde{x}^+ - x^+\|_Q^2 \leq \ \sigma(1+\kappa)\kappa^{\ell-1}\left(F(x^1 - x^\star, u^1 - u^\star) + F^\star(\lambda^1 - \lambda^\star)\right). \tag{4.38}$$

Furthermore, we apply Assumption 6 and the initialization procedure from Step 1 of Algorithm 4

$$F(x^1 - x^\star, u^1 - u^\star) + F^\star(\lambda^1 - \lambda^\star) \leq \gamma^2\|x_0\|_Q^2, \tag{4.39}$$

to get the upper bound in the following form

$$\|\tilde{x}^+ - x^+\|_Q^2 \leq 4\sigma\left(\frac{1+\kappa}{\kappa}\right)\kappa^\ell\gamma^2\|x_0\|_Q^2. \tag{4.40}$$

As presented in Fedorová et al. (2023), we consider two cases in our proof based on the norm $\|x_0\|_Q$.

Case 1: $\|x_0\|_Q > 1$. Substituting (4.40) into the bound (4.30) from Theorem 5, we obtain

$$|V(\tilde{x}^+) - V(x^+)| \leq 2\left(\eta_1\gamma\sqrt{\frac{\sigma(1+\kappa)}{\kappa}} + \eta_2\sigma\gamma^2\frac{1+\kappa}{\kappa}\right)\kappa^{\ell/2}\|x_0\|_Q^2. \tag{4.41}$$

In parallel, by applying the standard descent condition of the MPC value function under Assumption 7, we have

$$V(x^+) \leq V(x_0) - \|x_0\|_Q^2. \tag{4.42}$$

Following the proof in Chapter 3, we can modify inequality (4.42) into the suboptimal variant presented in (3.21). To ensure that the resulting inequality guarantees a decrease in the value function, we require

$$V(\tilde{x}^+) \leq V(x_0) - \beta\|x_0\|_Q^2 \tag{4.43}$$

to hold, which is ensured if the following condition is satisfied

$$\beta = 1 - 2\left(\eta_1\gamma\sqrt{\frac{\sigma(1+\kappa)}{\kappa}} + \eta_2\sigma\gamma^2\frac{1+\kappa}{\kappa}\right)\kappa^{\ell/2} > 0. \tag{4.44}$$

Solving for $\bar{\ell}$ leads to the bound

$$\bar{\ell} > \frac{2\log\left(2\eta_1\gamma\sqrt{\frac{\sigma(1+\kappa)}{\kappa}} + 2\eta_2\sigma\gamma^2\frac{1+\kappa}{\kappa}\right)}{\log(1/\kappa)}. \tag{4.45}$$

Case 2: $\|x_0\|_Q \leq 1$. In this case, Assumption 5 and the local neighborhood bound from Proposition 2 simplify (4.30) to

$$|V(\tilde{x}^+) - V(x^+)| \leq \frac{\eta_2}{2}\|\tilde{x}^+ - x^+\|_Q^2. \tag{4.46}$$

Substituting (4.40) yields

$$|V(\tilde{x}^+) - V(x^+)| \leq 2\eta_2\sigma\gamma^2\frac{1+\kappa}{\kappa}\kappa^\ell\|x_0\|_Q^2, \tag{4.47}$$

and consequently, to satisfy (4.43) we need to set

$$\bar{\ell} > \frac{\log\left(2\eta_2\sigma\gamma^2\frac{1+\kappa}{\kappa}\right)}{\log(1/\kappa)}. \tag{4.48}$$

Finally, to guarantee stability uniformly in both cases, we adopt a more conservative formula that yields a higher number of required iterations. By comparing the expressions for both cases, the more conservative bound is given by (4.31). Therefore, using this stopping criterion in Algorithm 4 ensures asymptotic stability of the closed-loop system, regardless of the initial condition value. □

The constants $\eta_1, \eta_2, \sigma, \gamma$, and $\kappa < 1$ presented in Theorem 5 depend solely on the problem data and are independent of the initial condition $x(t) \in \mathbb{X}$. Their interpretation, as well as procedures for computing or estimating them, were detailed in Chapter 3.

Although Theorem 5 guarantees a decrease in the value function, it does not establish closed-loop asymptotic stability in the presence of state and terminal constraints. This is because recursive feasibility cannot be guaranteed unless some specific techniques, such as soft constraints (Zeilinger et al., 2014) or constraint tightening (Parsi et al., 2022), are applied. However, these modifications can make assumptions required for the theoretical guarantees in Theorem 5 invalid.

Nonetheless, numerical simulations presented in Section 4.6 consistently demonstrate that the proposed controller stabilizes the closed-loop system even when state constraints are active, indicating strong practical performance.

4.3.3 Feasibility

The formulation adopted in Algorithm 4 extends the previously proposed strategy in Jiang et al. (2021) by explicitly incorporating both state and input constraints into the MPC framework. However, solving the resulting constrained optimization problem only approximately, due to the fixed number of iterations, can introduce feasibility issues. In particular, the suboptimal control input $u_0^{\bar{\ell}}$ may yield a predicted successor state $Ax(t) + Bu_0^{\bar{\ell}}$ that violates the predefined state constraints, i.e., $Ax(t) + Bu_0^{\bar{\ell}} \notin \mathbb{X}$.

To address this challenge, the initial condition $x_0 = x(t)$ is incorporated directly into the Lagrangian formulation through an associated dual variable λ_0. This approach introduces flexibility in handling initial state constraints, allowing the controller to generate stabilizing control inputs even when the exact feasibility cannot be guaranteed. The resulting structure resembles that of soft-constrained MPC schemes (Zeilinger et al., 2014), where certain constraint violations are penalized rather than strictly prohibited.

While this relaxed formulation does not inherently ensure recursive feasibility in all scenarios, it aligns well with the goals of fast, real-time implementation on embedded platforms. For applications requiring formal guarantees of recursive feasibility under suboptimal execution, advanced techniques such as contractive constraint tightening with terminal set modifications may be applied (Shi et al., 2022). Nonetheless, the proposed method provides a practical trade-off between feasibility, computational efficiency, and controller performance.

4.4 Implementation Details and Complexity

The implementation of the proposed control strategy is based on a two-phase approach. In the offline phase, the MPC problem is decomposed into $N + 1$ subproblems using the ALADIN coordination framework. We recall that all subproblems for stages $k \in \{0, \dots, N-1\}$ share the same structure, while the terminal stage $k = N$ is treated separately. Consequently, only two representative parametric QPs are solved offline using multi-parametric programming, resulting in two explicit control laws. These are then stored and used repeatedly during online execution.

In the online phase, each iteration of the control algorithm involves two main computational steps: (i) the parallel evaluation of explicit control laws for all prediction steps and (ii) the solution of the structured LQR-type QP using an analytical method. This process is repeated for a fixed number of iterations or until a stopping criterion is met.

The algorithm is designed for fast runtime performance and deterministic execution, making it suitable for real-time applications with strict timing constraints.

The most computationally demanding operations of Algorithm 4 are contained in Step 2, which involves solving decoupled and coupled quadratic programs. Therefore, this section focuses primarily on analyzing the implementation aspects of Step 2a) and Step 2b), with particular emphasis on computational and memory efficiency for real-time and embedded applications.

As previously described, the decoupled optimization problems in Step 2a) are solved offline by constructing explicit solution maps. These are generated using the enumeration-based multi-parametric quadratic programming (mp-QP) algorithm presented in Herceg et al. (2015). The complexity of this offline pre-processing step depends solely on the number of polytopic critical regions that define the PWA control laws, specifically

$$N_{\mathrm{R}} = \max\{N_{\mathrm{R},1}, N_{\mathrm{R},N}\}. \tag{4.49}$$

As discussed, by dualizing the initial condition, only two explicit solution maps are required. The quantities $N_{\mathrm{R},1}$ and $N_{\mathrm{R},N}$ denote the number of critical regions associated with these two maps. Importantly, this quantity is independent of the prediction horizon N, as shown in Table 4.1 and supported by results in Boyd and Vandenberghe (2004); Borrelli et al. (2017); Oravec et al. (2017).

Each explicit map is structured as a binary search tree (Tøndel et al., 2003). The trees are constructed offline with $\mathcal{O}(N_{\mathrm{R}}^2)$ computational complexity, after which the solution to each decoupled QP can be retrieved online with complexity $\mathcal{O}(\log_2(N_{\mathrm{R}}))$. Because evaluations of these explicit control laws can be done independently at each k-th step, the process is naturally parallelizable. In this work, parallelization is achieved using the `OpenMP API` (OpenMP Architecture Review Board, 2008) in `C`, which further accelerates Step 2a) without introducing sequential bottlenecks.

The memory requirements scale with the number of critical regions, as each region is represented by a finite set of affine inequalities. A key advantage of this implementation is that only two explicit solution maps must be stored, regardless of the prediction horizon: one for the intermediate steps $(x_1^\star, u_1^\star)(\theta_k^\ell)$, valid for $k \in \{0, \ldots, N-1\}$, and one for the terminal state $x_N^\star(\theta_N^\ell)$. As a result, the memory footprint of Step 2a) remains constant with respect to N, offering a substantial improvement over conventional explicit MPC techniques, where memory usage may grow exponentially with the horizon (Oravec et al., 2017). Moreover, by storing only these two specific maps, memory consumption is reduced by approximately 30% compared to the method proposed in Jiang et al. (2021).

In summary, the multi-parametric implementation strategy adopted in Step 2a) achieves fixed memory complexity and logarithmic runtime complexity per step, independent of the prediction horizon. These properties, combined with efficient parallelism, make the algorithm particularly well-suited for real-time implementation in embedded control environments.

In Step 2b) of Algorithm 4, the large-scale equality-constrained QP (4.8) is solved based on the local variables updates. Due to its purely equality-constrained structure, the problem reduces to solving a sparse and structured system of linear equations. Importantly, all matrices involved in (4.8) remain constant across online iterations. As a result, all necessary matrix factorizations and decompositions can be computed offline. To efficiently exploit the block-banded structure of the resulting KKT system, a Riccati recursion scheme can be employed. This leads to an offline computational complexity of at most $\mathcal{O}(Nn_{\mathrm{x}}^3)$ and an online cost of $\mathcal{O}(Nn_{\mathrm{x}}^2)$ (Bertsekas, 2012).

The computational and memory requirements for both the decoupled and coupled QP steps are summarized in Table 4.1. Note that Step 2a) is inherently parallelizable, enabling efficient execution across multiple cores or processing units.

Table 4.1: Computational and memory complexity of Steps 2a) and 2b) of Algorithm 4, considering fully parallelized online computations in Step 2a).

Step	Offline Complexity	Online Complexity	Memory Requirement
2a)	$\mathcal{O}(N_{\mathrm{R}}^2)$	$\mathcal{O}(N\log_2(N_{\mathrm{R}}))$	$\mathcal{O}(N_{\mathrm{R}})$
2b)	$\mathcal{O}(Nn^3)$	$\mathcal{O}(Nn^2)$	$\mathcal{O}(Nn^2)$

In summary, Algorithm 4 can be implemented by using static memory only, allocating at most $\mathcal{O}(N_{\mathrm{R}} + Nn^2)$ floating point numbers with $n = n_{\mathrm{x}} + n_{\mathrm{u}}$. Here, the explicit solution maps of the decoupled QPs in Step 2a), as well as the matrix factorizations associated with coupled QP in Step 2b), are precomputed offline. Because Theorem 5 provides an explicit formula for computing a constant number of iterations $\bar{\ell}$ such that a controller ensuring asymptotic stability of the closed-loop is obtained, the online runtime of Algorithm 4 is constant and of order $\mathcal{O}(N\log_2(N_{\mathrm{R}}) + Nn^2)$. Thus, Algorithm 4 is considered a real-time MPC method in the sense that it has a constant runtime and constant memory requirements for any given N. Its main advantage compared to existing MPC controllers is that it scales up easily for systems with long prediction horizons N.

4.5 Open-Source Toolbox

The proposed control strategy is implemented in the open-source `C`-code library `ParExMPC`[1]. The structure of `ParExMPC` follows the real-time control architecture described in Algorithm 4 and is tailored for efficient deployment on embedded or industrial-grade hardware. To ensure the provided open-source toolbox runs as intended, users must install the required toolboxes and dependencies. The `MATLAB` implementation of Algorithm 4 relies exclusively on open-source tools and solvers, as outlined below:

- **Step 2a) Evaluation of decoupled QPs**

  The decoupled optimization problems (4.6) are solved offline using multi-parametric programming. The solution maps are generated using solvers provided by the `MPT` toolbox, including `PLCP`, `enum`, and `mpqp`, which efficiently exploit the structure of each subproblem to produce explicit control laws.

- **Step 2b) Evaluation of the coupled QP**

  The coordination step (4.8), which aligns the local solutions, is formulated as a structured LQR-based QP. This problem is solved analytically, avoiding the need for iterative solvers and ensuring a predictable and low-latency implementation suitable for real-time applications.

In subsequent sections, we will present the implementation aspects of the proposed toolbox, with a focus on the automatic generation of `C` code from `MATLAB`. We show how parallelization is handled internally and how users can configure relevant parameters for parallel execution. In addition, a tutorial example is provided to illustrate the end-to-end workflow of applying the toolbox to a simple, controlled system.

4.5.1 C-Code Generation

The `ParExMPC` toolbox is written in the `C` programming language and provides a `MATLAB` interface for problem definition and prototyping. As shown in Listing 4.1, users can specify system dynamics via matrices A, B, and define the objective function using weighting matrices Q, R, and P. Additional optional arguments allow users to define reference trajectories, constraints, horizon length N, and scaling options.

[1] Project homepage: https://github.com/sujunyan/ParExMPC/wiki

Listing 4.1: MATLAB API for defining the control problem using ParExMPC toolbox.

```
% The constructor of the peMPC
% input:
%   A,B: the matrices that define the linear system dynamics.
%   Q,R,P: the matrices that define the cost function
% Optional input:
%   xr: the reference state. The default value is zero
%   xNr: the reference terminal state. The default value is zero
%   ur: the reference control input. The default value is zero
%   xmin,xmax: the state constraint. The default value is [-inf,inf]
%   umin,umax: the control input constraint. The default value is [-inf,inf]
%   N: the time horizon. The default value is 10
%   cons_mul: the constraint multiplier, often set to less than 1 to avoid
     constraint violation. The default value is 1
%   par_flag: the boolean flag to enable/disable the parallel computing. The
     default value is true
%   par_threshold: enable parallel computing if the time horizon is larger
     than the threshold. The default value is 20
% Output:
%   obj: the object of the peMPC
obj = peMPC(A,B,Q,R,P,{optional inputs})
```

Listing 4.2: MATLAB API for getting control input using pre-generated mex function in ParExMPC toolbox.

```
% Input:
%   x0: The initial state
%   max_iter: The maximum iteration
%   tol: The tolerance for the algorithm to terminate
%   x:  initial guess of primal variable
%   lam: initial guess of dual variable
% Output:
%   x: The shifted primal variable for warm-start
%   lam: The shifted dual variable for warm-start
%   u0: The control input computed by the peMPC controller
[x,lam,u0] = peMPC_controller_mex(x0,max_iter,tol,x,lam)
```

The toolbox supports parallel execution through a configuration flag `par_flag` and a user-defined threshold `par_threshold` that activates parallelization only if N exceeds a specified limit. This feature helps avoid unnecessary overhead in small-scale problems. Parallelization of Step 2a) is implemented using the `OpenMP` standard (OpenMP Architecture Review Board, 2008), specifically by applying the `omp parallel for` directive to loop-based evaluations of the explicit control maps. This requires compiler support for `OpenMP`, which is available in most modern `C` compilers.

Upon defining the control problem, the toolbox automatically generates a set of `C` header files containing all problem-specific data (e.g., system matrices, weighting

matrices, and precomputed Riccati recursion matrices). These files are compiled together with the source code into standalone binaries (e.g., `.mex` files) for use in real-time operation. The `MATLAB` API for evaluation of control input through `.mex` function is shown in Listing 4.2.

The implementation is designed to be self-contained and lightweight, making it suitable for resource-constrained embedded platforms. To further reduce dependencies, custom linear algebra routines have been implemented internally, offering competitive performance in experimental benchmarks. Alternatively, users can opt to link external BLAS libraries such as `OpenBLAS` (Xianyi et al., 2012) or `BLASFEO` (Frison et al., 2018) for enhanced numerical performance.

In the following subsection, we will present a tutorial example to illustrate the workflow of using the `ParExMPC` toolbox.

4.5.2 A Tutorial Example

The `ParExMPC` toolbox provides a user-friendly interface for modeling the MPC problem in (4.1) and obtaining its corresponding parallel real-time controller. We shall illustrate the `MATLAB` interface by the following tutorial example with time horizon $N = 3$ and no state constraints. Consider the MPC design problem in (4.1), where:

$$A = \begin{bmatrix} 0.7115 & -0.4345 \\ 0.4345 & 0.8853 \end{bmatrix}, \quad B = \begin{bmatrix} 0.2173 \\ 0.0573 \end{bmatrix}, \quad \text{s.t.} \quad u \in [-5, 5],$$
$$P = \begin{bmatrix} 20.2867 & 7.7432 \\ 7.7432 & 33.9237 \end{bmatrix}, \quad Q = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}, \quad R = 1.$$

The listing below gives the code sample on how to solve the tutorial example using the proposed `ParExMPC` toolbox. The sample includes three parts:

1. *Problem formulation:* Lines 1-9 define the system dynamics, cost function weighting matrices, and input constraints. The terminal penalty matrix P is calculated by the `MATLAB` built-in function `dlqr`, see Line 6. We also define prediction horizon N and the initial condition x_0 in this part, see Lines 8-9.

```
% define the problem ------
A = [0.7115,-0.4345; 0.4345, 0.8853];
B = [0.2173;0.0573];
Q = [10,0;0,10];
R = 1;
[~,P] = dlqr(A,B,Q,R);
umin = -5; umax = 5;
N = 3;
x0 = [10;0];
```

2. *Code generation and compilation:* Lines 12-13 illustrate the main interface of the `ParExMPC` Toolbox for code generation and compilation.

```
% use the ParExMPC interface ------
% call help peMPC to see available optional parameters.
mpc0 = peMPC(A,B,Q,R,P,'umin',umin,'umax',umax,'N',N);
mpc0 = mpc0.build;
```

In line 12, we construct the object of the main class called `peMPC` with the problem parameters we previously defined. To see a complete list of parameters, one can call a command: "`help peMPC`" in `MATLAB`. In line 13, we execute the `build` method. It is a bundle method that initializes the object, generates a `C`-code, and compiles it into a `.mex` file. In particular, the `build` method includes the following steps:

I) initialization of the `peMPC` object by computing the factorized matrix for Riccati recursion;

II) pre-computation of the solution maps (4.7a)–(4.7b) with `MPT` toolbox (Herceg et al., 2013);

III) transformation of the system matrices A, B, Q, R, P and the factorized matrices into the `C`-format;

IV) compilation the `C`-code into a `.mex` file, which will be used in the next part by invoking the `MATLAB` compiler.

3. *Usage of MEX function:* Lines 22-36 illustrate how to use the generated `.mex` file in the closed-loop simulation of MPC.

```
14 % start of MPC simulation ------
15 maxiter = 5;
16 nsim = 100;
17 [nx,nu] = size(B);
18 % the tolerance in the MPC iteration
19 tol = 1e-04;
20 X = [];
21 U = [];
22 for i = 1:nsim
23 % solve the first problem with a large maximum iteration
24 if (i == 1)
25 lam = zeros((N+1)*nx,1);
26 z = zeros(N*(nx+nu) + nx ,1);
27 [z,lam,u0] = peMPC_controller_mex(x0,100,tol,z,lam);
28 else
29 [z,lam,u0] = peMPC_controller_mex(x0,maxiter,tol,z,lam);
30 end
31 % simulate one step of the system dynamics
32 x0 = A*x0+B*u0;
33 % store control profiles
34 X = [X,x0];
35 U = [U,u0];
36 end
37 % plot control profiles
38 figure (1), hold on, stairs([1:100],X(1,:)), stairs([1:100],X(2,:))
39 figure (2), stairs([1:100],U)
```

Here, x_0 is the initial state of the current step, `maxiter` is the maximum number of iterations for the real-time computation, `z`, `lam` are the initial guesses of the stacked primal variable (x^1, u^1) and the dual variable λ^1, respectively. For the first iteration, we set `z` and `lam` to zero vectors and use a large `maxiter` to obtain an optimal point (line 27). In the following iterations, we use `zout` and `lamout` from the last iteration as initial guesses for warm-starting with small `maxiter` (line 29).

4.6 Case Studies

To evaluate the proposed real-time parallelizable MPC approach presented in this chapter, we first conduct a general performance comparison of the developed `ParExMPC` toolbox against several state-of-the-art solvers across a range of well-established benchmark systems. In addition, we demonstrate the applicability of `ParExMPC` on a resource-constrained embedded platform, specifically, the Arduino Uno, highlighting its efficiency under tight computational and memory limitations.

4.6.1 Numerical Examples

To thoroughly assess the computational efficiency and control performance of the proposed `ParExMPC` toolbox, we conducted a series of numerical experiments based on nine publicly available benchmark systems for validation of MPC problem formulations[2]. All simulations were executed in `MATLAB 2020b` on a computer with an AMD Ryzen 7 PRO 4750U processor with eight physical cores and 16 GB of RAM.

The proposed toolbox was evaluated alongside several well-established QP solvers, including:

- `OSQP` (Stellato et al., 2020): Operator splitting solver for quadratic programs.
- `Gurobi` (Gurobi Optimization, Inc., 2023): Commercial solver based on interior-point and simplex methods.
- `DAQP` (Arnström et al., 2022): Dual active-set solver designed for fast MPC applications.
- `quadprog` (MathWorks, Inc., 2023): `MATLAB` built-in solver using active-set and interior-point methods.
- `qpOASES` (Ferreau et al., 2014): Online active-set strategy for real-time applications.
- `MOSEK` (MOSEK ApS, 2023): Commercial conic and QP solver with high precision.
- `HPIPM` (Frison and Diehl, 2020): High-performance interior-point method optimized for embedded MPC.

Each benchmark is defined by a set of parameters characterizing system dynamics and constraint settings. These include the prediction horizon N, the number of states n_{x}, the number of control inputs n_{u}, and the number of state and input constraints per time step, denoted as $n_{\mathrm{b_x}}$ and $n_{\mathrm{b_u}}$, respectively. A scalar constraint multiplier, `cons_mul`, is employed to ensure feasibility during closed-loop execution, particularly for state-constrained systems.

The tables 4.2–4.3 summarize benchmarks configuration, the accuracy of the generated solution, average execution times from closed-loop simulations, and corresponding speedup ratios with respect to the solvers mentioned above that were obtained using

[2] https://github.com/ferreau/mpcBenchmarking

both serial (single-threaded) and parallel (multi-threaded via `OpenMP`) variants of the `ParExMPC` toolbox.

The results in Tables 4.2–4.3 show that for small-scale systems, `daqp` performs competitively. However, `ParExMPC` still maintains a speed advantage of at least 1.2× with same solution accuracy. For large-scale problems, `ParExMPC` achieves a minimum speedup of 1.1× even over specialized solvers such as `OSQP`, which are optimized for high-dimensional QPs.

Notably, the `chain` benchmark posed significant challenges for solvers `daqp` and `qpOASES`, both of which failed to return results in a reasonable time frame. This aligns with the theoretical expectations, given they exhibit cubic computational complexity.

It is worth emphasizing that `ParExMPC` achieves this performance by evaluating precomputed explicit control laws. The offline generation of these maps typically takes between 2 and 40 seconds, depending on system complexity. While the re-generation of explicit maps is required when system dynamics change, it remains negligible compared to the speedups produced during online control.

To evaluate solution quality, we consider the cumulative closed-loop cost

$$J = \sum_{i=0}^{\infty} \ell(x_i^{\mathrm{cl}}, u_i^{\mathrm{cl}}),$$

where x_i^{cl} and u_i^{cl} denote the state and control input at time step i. For each solver, the cost ratio and relative deviation with respect to `ParExMPC` are defined as

$$\varepsilon_{\mathrm{ratio}} = \frac{J_{\mathrm{ParExMPC}}}{J_{\mathrm{SOLVER}}}, \qquad \varepsilon_{\mathrm{error}} = \frac{J_{\mathrm{ParExMPC}} - J_{\mathrm{SOLVER}}}{J_{\mathrm{SOLVER}}}.$$

These metrics allow direct comparison of control performance. In all evaluated benchmarks, `ParExMPC` maintains a level of solution accuracy comparable to that of other solvers, even in cases where it achieves faster runtimes. In particular, the deviations in $\varepsilon_{\mathrm{error}}$ remain negligible for most problems, confirming the numerical reliability of the proposed distributed-explicit control strategy.

The results demonstrate that the parallel execution mode of `ParExMPC` is particularly beneficial for large-scale systems, where the computational workload significantly outweighs the overhead introduced by `MATLAB`'s parallelization framework. In contrast, for small-scale problems, this overhead becomes non-negligible, and serial execution proves to be more efficient. Nevertheless, even in serial mode, `ParExMPC` consistently outperforms several state-of-the-art solvers.

Table 4.2: Comparison of control performance ($\varepsilon_{\text{error}}$) and solver times between the `ParExMPC` toolbox and established implicit MPC solvers across various benchmark systems.

name	toy	fiordos	forces	pendulum	DC motor	robotArm	CSTR	helicopter	chain
n_{x}	2	2	2	3	4	4	4	6	57
$n_{\text{b}_{\text{x}}}$	–	–	2	–	2	2	–	2	–
n_{u}	1	1	1	1	1	2	2	2	3
$n_{\text{b}_{\text{u}}}$	1	1	1	1	1	2	2	2	3
N	10	5	9	15	20	20	80	100	100
$\bar{\ell}$	5	5	5	5	5	5	5	20	20
`cons_mul`	1	1	1	1	1	0.85	1	0.95	1
				$\varepsilon_{\text{error}}$					
`OSQP`	$4.62 \cdot 10^{-16}$	$-1.79 \cdot 10^{-16}$	$1.15 \cdot 10^{-7}$	$4.96 \cdot 10^{-5}$	$-3.95 \cdot 10^{-7}$	$2.02 \cdot 10^{-2}$	$-2.36 \cdot 10^{-2}$	$-3.82 \cdot 10^{-3}$	$-1.28 \cdot 10^{-1}$
`Gurobi`	$3.70 \cdot 10^{-13}$	$-3.07 \cdot 10^{-12}$	$1.15 \cdot 10^{-7}$	$4.96 \cdot 10^{-5}$	$-1.74 \cdot 10^{-12}$	$2.02 \cdot 10^{-2}$	$-2.36 \cdot 10^{-2}$	$-3.82 \cdot 10^{-3}$	$-1.28 \cdot 10^{-1}$
`daqp`	$9.71 \cdot 10^{-15}$	$-1.37 \cdot 10^{-12}$	$1.15 \cdot 10^{-7}$	$4.96 \cdot 10^{-5}$	$-1.96 \cdot 10^{-10}$	$2.02 \cdot 10^{-2}$	$-2.36 \cdot 10^{-2}$	$-3.82 \cdot 10^{-3}$	†
`quadprog`	$-1.95 \cdot 10^{-9}$	$-1.23 \cdot 10^{-9}$	$1.12 \cdot 10^{-7}$	$4.91 \cdot 10^{-5}$	$-4.43 \cdot 10^{-10}$	$2.02 \cdot 10^{-2}$	$-2.36 \cdot 10^{-2}$	$-3.82 \cdot 10^{-3}$	$-1.28 \cdot 10^{-1}$
`qpOASES`	$1.54 \cdot 10^{-16}$	$-3.59 \cdot 10^{-16}$	$1.15 \cdot 10^{-7}$	$4.96 \cdot 10^{-5}$	$-2.68 \cdot 10^{-11}$	$2.02 \cdot 10^{-2}$	$-2.36 \cdot 10^{-2}$	$-3.82 \cdot 10^{-3}$	†
`MOSEK`	$-1.95 \cdot 10^{-9}$	$-1.23 \cdot 10^{-9}$	$1.12 \cdot 10^{-7}$	$4.91 \cdot 10^{-5}$	$-4.43 \cdot 10^{-11}$	$2.02 \cdot 10^{-2}$	$-2.36 \cdot 10^{-2}$	$-3.82 \cdot 10^{-3}$	$-1.28 \cdot 10^{-1}$
`HPIPM`	$-1.28 \cdot 10^{-10}$	$8.68 \cdot 10^{-6}$	$-7.76 \cdot 10^{-6}$	$-8.41 \cdot 10^{-1}$	$-9.51 \cdot 10^{-1}$	$1.69 \cdot 10^{-2}$	$-1.37 \cdot 10^{-3}$	$5.77 \cdot 10^{-3}$	$-7.68 \cdot 10^{-1}$
				solver time in miliseconds					
`ParExMPC`									
non-parallel:	$6.609 \cdot 10^{-5}$	$6.890 \cdot 10^{-5}$	$8.041 \cdot 10^{-5}$	$6.068 \cdot 10^{-5}$	$7.601 \cdot 10^{-4}$	$7.109 \cdot 10^{-4}$	$8.221 \cdot 10^{-4}$	$3.849 \cdot 10^{-3}$	$1.440 \cdot 10^{-1}$
parallel:	$7.925 \cdot 10^{-4}$	$9.993 \cdot 10^{-4}$	$8.094 \cdot 10^{-4}$	$5.403 \cdot 10^{-4}$	$8.154 \cdot 10^{-4}$	$6.843 \cdot 10^{-4}$	$9.901 \cdot 10^{-4}$	$3.519 \cdot 10^{-3}$	$9.940 \cdot 10^{-2}$
`OSQP`	$1.228 \cdot 10^{-3}$	$1.091 \cdot 10^{-3}$	$1.096 \cdot 10^{-3}$	$1.375 \cdot 10^{-3}$	$1.578 \cdot 10^{-3}$	$1.561 \cdot 10^{-3}$	$2.500 \cdot 10^{-3}$	$4.432 \cdot 10^{-3}$	$1.053 \cdot 10^{-1}$
`Gurobi`	$2.291 \cdot 10^{-3}$	$2.020 \cdot 10^{-3}$	$2.168 \cdot 10^{-3}$	$2.772 \cdot 10^{-3}$	$3.057 \cdot 10^{-3}$	$3.280 \cdot 10^{-3}$	$6.821 \cdot 10^{-3}$	$1.229 \cdot 10^{-2}$	$2.384 \cdot 10^{-1}$
`daqp`	$1.418 \cdot 10^{-4}$	$7.934 \cdot 10^{-5}$	$1.268 \cdot 10^{-4}$	$8.586 \cdot 10^{-4}$	$1.982 \cdot 10^{-3}$	$4.362 \cdot 10^{-3}$	$1.649 \cdot 10^{-3}$	$4.343 \cdot 10^{-3}$	†
`quadprog`	$4.910 \cdot 10^{-3}$	$5.133 \cdot 10^{-3}$	$5.064 \cdot 10^{-3}$	$5.770 \cdot 10^{-3}$	$9.739 \cdot 10^{-3}$	$6.647 \cdot 10^{-3}$	$1.095 \cdot 10^{-2}$	$1.826 \cdot 10^{-2}$	$4.381 \cdot 10^{-1}$
`qpOASES`	$1.190 \cdot 10^{-3}$	$5.646 \cdot 10^{-4}$	$1.178 \cdot 10^{-3}$	$6.981 \cdot 10^{-3}$	$4.707 \cdot 10^{-2}$	$6.340 \cdot 10^{-2}$	$7.020 \cdot 10^{-2}$	$4.290 \cdot 10^{-1}$	†
`MOSEK`	$1.985 \cdot 10^{-3}$	$1.818 \cdot 10^{-3}$	$2.115 \cdot 10^{-3}$	$2.762 \cdot 10^{-3}$	$5.868 \cdot 10^{-3}$	$4.108 \cdot 10^{-3}$	$7.527 \cdot 10^{-3}$	$7.127 \cdot 10^{-3}$	$3.652 \cdot 10^{-2}$
`HPIPM`	$1.263 \cdot 10^{-4}$	$1.110 \cdot 10^{-4}$	$1.140 \cdot 10^{-4}$	$1.782 \cdot 10^{-4}$	$1.389 \cdot 10^{-4}$	$4.521 \cdot 10^{-4}$	$8.253 \cdot 10^{-4}$	$2.822 \cdot 10^{-3}$	$9.664 \cdot 10^{-2}$

Table 4.3: Comparison of cost ratio ($\varepsilon_{\text{ratio}}$) and runtime speedup between the `ParExMPC` toolbox and established implicit MPC solvers across various benchmark systems.

name	toy	fiordos	forces	pendulum	DC motor	robotArm	CSTR	helicopter	chain
n_{x}	2	2	2	3	4	4	4	6	57
$n_{\text{b}_{\text{x}}}$	-	-	2	-	2	2	-	2	-
n_{u}	1	1	1	1	1	2	2	2	3
$n_{\text{b}_{\text{u}}}$	1	1	1	1	1	2	2	2	3
N	10	5	9	15	20	20	80	100	100
$\bar{\ell}$	5	5	5	5	5	5	5	20	20
`cons_mul`	1	1	1	1	1	0.85	1	0.95	1
					$\varepsilon_{\text{ratio}}$				
`OSQP`	1.00	1.00	1.00	1.00	1.00	1.02	0.98	1.00	0.87
`Gurobi`	1.00	1.00	1.00	1.00	1.00	1.02	0.98	1.00	0.87
`daqp`	1.00	1.00	1.00	1.00	1.00	1.02	0.98	1.00	†
`quadprog`	1.00	1.00	1.00	1.00	1.00	1.02	0.98	1.00	0.87
`qpOASES`	1.00	1.00	1.00	1.00	1.00	1.02	0.98	1.00	†
`MOSEK`	1.00	1.00	1.00	1.00	1.00	1.02	0.98	1.00	0.87
`HPIPM`	1.00	1.00	1.00	0.16	0.05	1.02	0.98	1.01	0.23
					speedup				

parallelization:	no	yes	no	yes	no	yes	no	yes	no	yes	no	yes	no	yes	no	yes	no	yes
`OSQP`	18.6	1.5	15.8	1.1	13.6	1.4	22.7	2.5	2.1	1.9	2.2	2.3	3.0	2.5	1.2	1.3	0.7	1.1
`Gurobi`	34.7	2.9	29.3	2.0	27.0	2.7	45.7	5.1	4.0	3.7	4.6	4.8	8.3	6.9	3.2	3.5	1.7	2.4
`daqp`	2.1	0.2	1.2	0.1	1.6	0.2	14.1	1.6	2.6	2.4	6.1	6.4	2.0	1.7	1.1	1.2	†	†
`quadprog`	74.3	6.2	74.5	5.1	63.0	6.3	95.1	10.7	12.8	11.9	9.4	9.7	13.3	11.1	4.7	5.2	3.0	4.4
`qpOASES`	18.0	1.5	8.2	0.6	14.6	1.5	115.0	12.9	61.9	57.7	89.2	92.7	85.4	70.9	111.5	121.9	†	†
`MOSEK`	30.0	2.5	26.4	1.8	26.3	2.6	45.5	5.1	7.7	7.2	5.8	6.0	9.2	7.6	1.9	2.0	2.5	3.7
`HPIPM`	1.9	0.2	1.6	0.1	1.4	0.1	2.9	0.3	0.2	0.2	0.6	0.7	1.0	0.8	0.7	0.8	0.7	1.0

To evaluate the performance of the proposed method in a practical closed-loop setting, we consider the `helicopter` benchmark model introduced in Tøndel and Johansen (2002), characterized by $n_x = 6$ states, $n_u = 2$ control inputs, and a prediction horizon of $N = 100$. This corresponds to the `helicopter` benchmark entry in Tables 4.2–4.3.

The following state-space matrices define the discrete-time linear system

$$A = \begin{bmatrix} 0.99 & 0 & 0.01 & 0 & 0 & 0 \\ 0 & 0.99 & 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0.99 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.99 & 0 & 0 \\ 0.01 & 0 & 0 & 0 & 0.99 & 0 \\ 0 & 0.01 & 0 & 0 & 0 & 0.99 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 \\ 0.0001 & -0.0001 \\ 0.0019 & -0.0019 \\ 0.0132 & -0.0132 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

The cost function used for optimization is quadratic, with state and input weighting matrices given by

$$Q = \mathrm{diag}(100, 100, 10, 10, 400, 200), \qquad R = \mathrm{diag}(0.001, 0.001).$$

The terminal cost matrix P is computed via the discrete-time algebraic Riccati equation using `dlqr` function in `MATLAB`. The following constraints are enforced:

$$-1 \leq u_1 \leq 3, \quad -1 \leq u_2 \leq 3, \qquad -0.44 \leq x_3 \leq 0.44, \quad -0.6 \leq x_4 \leq 0.6.$$

Figure 4.2 presents a comparison of the cumulative running cost J_k and resulting state trajectories obtained using the proposed `ParExMPC` toolbox and the `OSQP` solver, selected as the most substantial competing method in terms of performance. The running cost at time step k is computed as

$$J_k = \sum_{i=0}^{k} \ell(x_i^{\mathrm{cl}}, u_i^{\mathrm{cl}}),$$

where x_i^{cl} and u_i^{cl} are the closed-loop state and control input, respectively. As shown in Table 4.3, `ParExMPC` achieves a 1.3× speedup over `OSQP`, with negligible control performance loss.

Further performance comparisons across solvers are illustrated in Figure 4.3, where the runtime is plotted as a function of prediction horizon N. On the left-hand plot (logarithmic scale), `qpOASES` shows exponential growth in runtime with respect to N, consistent with its cubic worst-case complexity. In contrast, both `ParExMPC` and `OSQP` exhibit nearly linear complexity. The right-hand plot (linear scale) omits `qpOASES` due

to scale mismatch. However, it highlights the considerably lower complexity rate of `ParExMPC` compared to `OSQP`, supporting its suitability for real-time applications.

Finally, Figure 4.4 shows how the closed-loop cost varies with the prediction horizon N and the maximum number of solver iterations $\bar{\ell}$. As expected, longer horizons lead to better approximations of the infinite-horizon cost J_∞, thereby improving control performance.

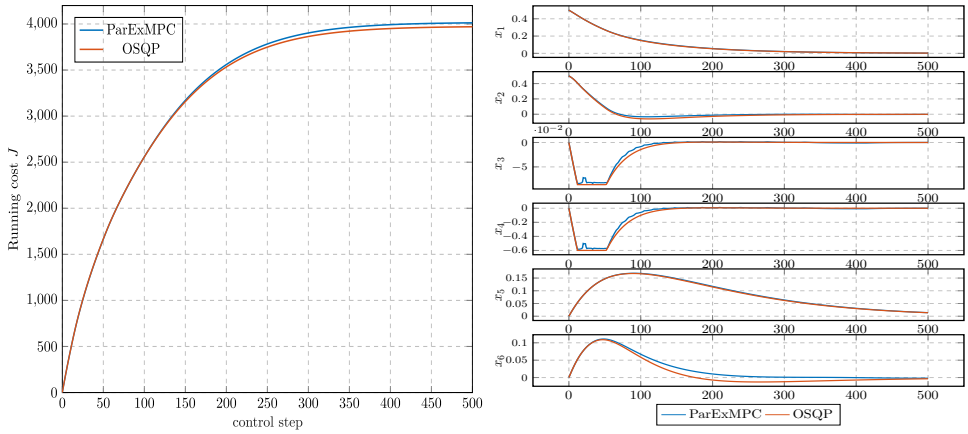


Figure 4.2: Comparison of `ParExMPC` and `OSQP` on the helicopter benchmark system with $\bar{\ell} = 100$ and a prediction horizon of $N = 20$. The left plot shows the cumulative running cost J, while the right plot depicts the closed-loop evolution of the state variables for both solvers at each control step.

4.6.2 Implementation on Embedded Hardware

One of the main goals behind the development of the `ParExMPC` toolbox is to enable its implementation on embedded hardware. In general, to use the `ParExMPC` toolbox on resource-constrained platforms, the explicit solutions must first be generated on suitable hardware. These solutions can then be deployed on the embedded platform, where they can be evaluated efficiently without the need for any external solver. In this subsection, we discuss the implementation capabilities of the proposed real-time MPC controller on such platforms.

As a representative test case, we deploy the generated controller on an Arduino Uno with the ATmega328 microchip (16 MHz clock, 2 kB SRAM, 32 kB Flash memory). This microcontroller is known for its extremely limited resources, posing a significant challenge for traditional explicit MPC implementations that require to store large

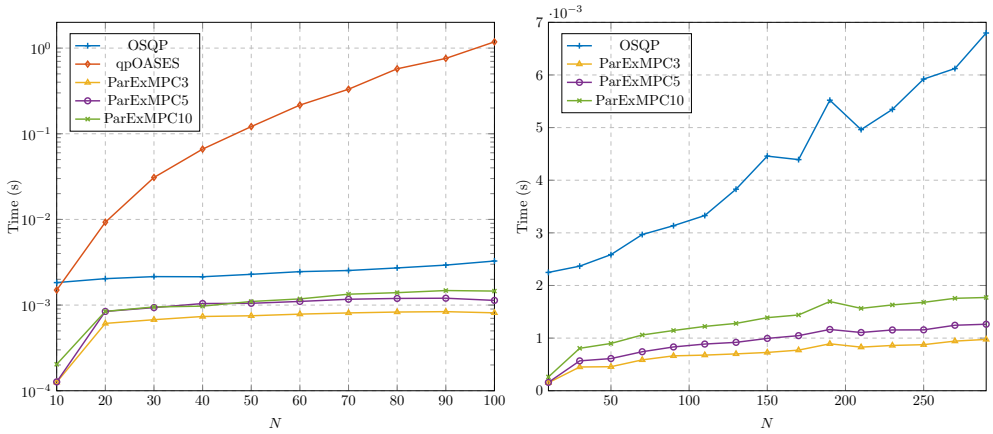


Figure 4.3: Runtime comparison for different settings across varying prediction horizons N for the helicopter benchmark system with $n_x = 6$ and $n_u = 2$. The left plot uses a logarithmic scale, while the right plot uses a linear scale for improved clarity. The trajectories labeled "ParExMPC3", "ParExMPC5", "ParExMPC10" correspond to maximum iteration counts of 3, 5, and 10, respectively, as evaluated using the `ParExMPC` toolbox.

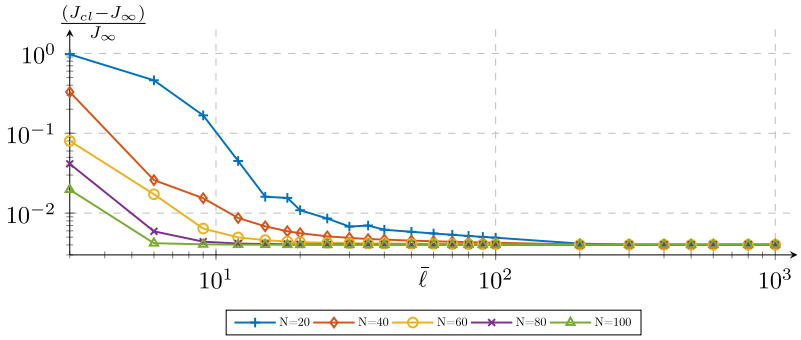


Figure 4.4: Visualization of the dependence of relative closed-loop performance on the maximum number of iterations $\bar{\ell}$ with respect to the optimal objective function J_∞ for the helicopter benchmark case ($n_\mathrm{x} = 6$, $n_\mathrm{u} = 2$). Different curves correspond to different prediction horizons N.

region partitions and solver metadata.

For this demonstration, we consider a low-dimensional control problem formulated in (4.1) with a short prediction horizon $N = 3$. The system dynamics and cost parameters are specified as

$$A = \begin{bmatrix} 0.7115 & -0.4345 \\ 0.4345 & 0.8853 \end{bmatrix}, \quad B = \begin{bmatrix} 0.2173 \\ 0.0573 \end{bmatrix},$$
$$Q = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}, \quad R = 1, \quad P = \begin{bmatrix} 20.2867 & 7.7432 \\ 7.7432 & 33.9237 \end{bmatrix},$$
$$u \in [-5, 5].$$

The terminal cost matrix P is obtained using the `dlqr` function in MATLAB.

Compared to conventional explicit MPC approaches that struggle with memory overhead, particularly when deployed on the ATmega328's 2 kB of SRAM, the code generated by `ParExMPC` compiles and runs successfully. Total memory usage was only 1,468 bytes (approximately 71% of the available SRAM), leaving sufficient headroom for additional computations.

The measured runtime of the control scheme on the Arduino Uno does not exceed 35 milliseconds, achieving approximately 30 Hz sampling rate. This demonstrates the practical applicability of `ParExMPC` in real-time embedded scenarios, even on extremely limited platforms.

4.7 Discussion

This chapter presented a novel MPC framework designed for linear systems with state and input constraints. The proposed approach explicitly addresses the computational and memory limitations common in embedded and industrial control applications. The method integrates concepts from distributed optimization, explicit MPC, and parallel computing into a unified real-time control strategy.

The resulting framework is implemented in the open-source toolbox `ParExMPC`, which generates lightweight, self-contained `C` code suitable for deployment on memory-constrained hardware as well. Numerical experiments conducted on a set of benchmark systems confirm the method's practical viability. Compared to several state-of-the-art quadratic programming solvers, including `OSQP`, `qpOASES`, and `daqp`, `ParExMPC` consistently delivers significant runtime speedups, ranging from 1.2× to over 3×, without compromising control performance.

Additionally, the method was successfully deployed on an embedded platform, specifically the Arduino Uno. This highlights the ability of `ParExMPC` to bridge the gap between advanced control strategies and real-time implementations on embedded hardware.

Although the proposed approach does not provide formal asymptotic stability guarantees in the presence of state constraints, the derived theoretical bounds on the required number of solver iterations offer predictability and robustness. Extensive simulation results across all tested scenarios support the empirical stability and performance of the controller. Together with the established convergence guarantee for the fixed-iteration formulation, the framework lays a solid foundation for scalable and certifiable real-time MPC implementations.

Chapter 5

A Real-Time Parallelizable MPC for Large-scale Systems

This chapter introduces a distributed real-time MPC scheme tailored for large-scale, network-structured linear systems with polyhedral state and input constraints. The proposed method builds on the ALADIN optimization framework. We prove that the ALADIN algorithm with fixed scaling matrices converges to the global optimum. Furthermore, we extend the algorithm to operate with a fixed number of iterations per control step, ensuring predictable computational complexity and making it suitable for real-time implementation. A central component of the approach is the use of a generalized stopping criterion, initially developed in Chapter 3, which guarantees closed-loop stability for solvers exhibiting linear convergence. In the present context, this criterion is explicitly adapted to the ALADIN algorithm and extended to accommodate logarithmic barrier functions for handling inequality constraints. This adaptation allows us to establish theoretical guarantees even in the presence of state constraints. Furthermore, we decompose the MPC problem simultaneously along the prediction horizon (temporal domain) and across the state space (spatial domain), resulting in even smaller subproblems than those presented in Chapter 4. This decomposition enables efficient parallelization and significantly reduces the complexity of evaluating local solutions. The resulting formulation yields a lightweight, theoretically sound, and computationally efficient MPC controller. The results demonstrate the potential of the proposed approach to compete with well-established solvers while eliminating the dependency on external toolboxes or solvers. Its low computational and memory requirements make it suitable for real-time implementation. Furthermore, the use of spatio-temporal decomposition makes the algorithm well-suited for network-structured systems, where the global solution is obtained through cooperation among subproblems distributed across the network. The results presented in this chapter are based on the publication Jiang, Fedorová, Schwan, Oravec, and Jones (2025a), currently under review.

5.1 Distributed Convex Nonlinear Programming

To develop a distributed MPC framework, we adopt the ALADIN algorithm, as outlined in Algorithm 2. This method demonstrates superior performance compared to other established approaches, particularly by providing a linear convergence rate even in the worst-case scenario (Houska and Jiang, 2021). However, before proceeding, it is necessary to establish the global linear convergence of Algorithm 2 with fixed scaling matrices, as this result has not yet been proven in the literature.

To proceed with the convergence analysis, we first introduce the following assumption.

Assumption 8. *We assume that:*

a) Each local objective function f_i in (2.14) is closed, proper, and twice continuously differentiable. Furthermore, f_i is strongly convex with a Lipschitz continuous gradient, i.e., there exist positive definite matrices $\underline{H}_i, \overline{H}_i \succ 0$ such that

$$\underline{H}_i \preceq \nabla^2 f_i(z) \preceq \overline{H}_i \quad \text{for all } z;$$

b) The coupling matrix $D = [D_1, \ldots, D_M]$ in (2.10) has full row rank, i.e., the Linear Independence Constraint Qualification (LICQ) holds.

Based on Assumption 8 a), there exists a positive constant $c_{\mathrm{L}} > 0$ such that the gradient of the objective function in (2.10) satisfies the following Lipschitz continuity condition

$$\|\nabla f(z_1) - \nabla f(z_2)\|_2 \leq c_{\mathrm{L}} \cdot \|z_1 - z_2\|_H \,, \tag{5.1}$$

where the norm $\|\cdot\|_H$ is defined with respect to the block-diagonal matrix $H = \mathrm{diag}(H_i) \succ 0$.

To evaluate the distance between the current iterate of the algorithm and the optimal solution, we define the following Lyapunov-type function

$$\Omega(z, \lambda) = \|z - z^\star\|_H^2 + \|\lambda - \lambda^\star\|_\Sigma^2 \,, \tag{5.2}$$

where $\Sigma = DH^{-1}D^\top \succ 0$, and $(z^\star, \lambda^\star)$ denotes the primal-dual optimal solution of the distributed problem (2.10). We now recall the following results from the literature.

Lemma 6 (Theorem 1 in Houska et al. (2017))**.** *Let Assumption 8 hold. Then, the iterates of Algorithm 2 satisfy the following inequality*

$$\Omega(\bar{z}^{\ell+1}, \lambda^{\ell+1}) \leq \Omega(\bar{z}^{\ell}, \lambda^{\ell}) - 4\Phi(z^{\ell} - z^\star), \tag{5.3}$$

with the function $\Phi(\cdot) = \|\cdot\|_{\underline{H}}^2$.

This lemma establishes a strict descent property of the Lyapunov function Ω along the sequence of iterates generated by Algorithm 2. Detailed proof can be found in Houska et al. (2017).

The following theorem then establishes the global linear convergence of the Algorithm 2.

Theorem 7. *Let Problem* (2.10) *be feasible and Assumption 8 hold. Then, the primal-dual iterates* $(\bar{z}^\ell, \lambda^\ell)$ *generated by Algorithm 2 converge to the global optimum* $(z^\star, \lambda^\star)$ *with a linear convergence rate. In particular, there exists a constant* $\alpha > 0$ *such that the convergence rate*

$$\kappa = \frac{\alpha^2}{4 + \alpha^2} < 1 \tag{5.4}$$

satisfies the inequality

$$\Omega(\bar{z}^{\ell+1}, \lambda^{\ell+1}) \leq \kappa \cdot \Omega(\bar{z}^\ell, \lambda^\ell). \tag{5.5}$$

Proof. To prove the linear convergence condition in (5.5), we aim to show that there exists a constant $\alpha > 0$ such that

$$\Omega(\bar{z}^{\ell+1}, \lambda^{\ell+1}) \leq \alpha^2 \cdot \Phi(z^\ell - z^\star). \tag{5.6}$$

We begin by defining the optimality conditions for the coupled QP step (2.15) in Algorithm 2 in the form

$$\lambda^{\ell+1} = \lambda^\ell + \Sigma^{-1}(2Dz^\ell - D\bar{z}^\ell - e), \tag{5.7a}$$
$$\bar{z}^{\ell+1} = 2z^\ell - \bar{z}^\ell - H^{-1}D^\top(\lambda^{\ell+1} - \lambda^\ell), \tag{5.7b}$$

where $\Delta\lambda = \lambda^{\ell+1} - \lambda^\ell$. Next, we substitute the first-order optimality condition of the decoupled problem (2.14) given by

$$\begin{aligned} \nabla f(z^\ell) + D^\top\lambda^\ell + H(z^\ell - \bar{z}^\ell) = 0 \\ \implies \bar{z}^\ell = z^\ell + H^{-1}(\nabla f(z^\ell) + D^\top\lambda^\ell) \end{aligned} \tag{5.8}$$

into the equations above, yielding an equivalent representation as

$$\lambda^{\ell+1} = \Sigma^{-1}\left(Dz^\ell - DH^{-1}\nabla f(z^\ell) - e\right), \tag{5.9a}$$
$$\bar{z}^{\ell+1} = z^\ell - H^{-1}\left(\nabla f(z^\ell) + D^\top\lambda^{\ell+1}\right). \tag{5.9b}$$

We now derive upper bounds for the two terms comprising the Lyapunov function $\Omega(\bar{z}^{\ell+1}, \lambda^{\ell+1})$. Using (5.9a), we can bound term $\|\lambda^{\ell+1} - \lambda^\star\|_\Sigma^2$ by

$$\begin{aligned} \|\lambda^{\ell+1} - \lambda^\star\|_\Sigma &= \left\|\Sigma^{-1}D\left(z^\ell - z^\star - H^{-1}(\nabla f(z^\ell) - \nabla f(z^\star))\right)\right\|_\Sigma \\ &\leq \left\|\Sigma^{-1}D(z^\ell - z^\star)\right\|_\Sigma + \left\|\Sigma^{-1}DH^{-1}(\nabla f(z^\ell) - \nabla f(z^\star))\right\|_\Sigma. \end{aligned} \tag{5.10}$$

By expressing the first term in a symmetric form and applying the Lipschitz condition (5.1) to the second term, we obtain

$$\begin{aligned}\|\lambda^{\ell+1}-\lambda^\star\|_\Sigma &\le \left\|H^{-1/2}D^\top\Sigma^{-1}DH^{-1/2}\right\|_2\cdot\|z^\ell-z^\star\|_H\\ &\quad + c_L\cdot\left\|H^{-1}D^\top\Sigma^{-1}DH^{-1}\right\|_2\cdot\|z^\ell-z^\star\|_H\\ &= \alpha_1\cdot\|z^\ell-z^\star\|_H,\end{aligned}\tag{5.11}$$

where

$$\alpha_1=\left\|H^{-1/2}D^\top\Sigma^{-1}DH^{-1/2}\right\|_2+c_L\cdot\left\|H^{-1}D^\top\Sigma^{-1}DH^{-1}\right\|_2.\tag{5.12}$$

Now, using (5.9b), we derive the upper bound of $\|\bar{z}^{\ell+1}-z^\star\|_H$ as

$$\begin{aligned}\|\bar{z}^{\ell+1}-z^\star\|_H &= \left\|z^\ell-z^\star-H^{-1}\left(\nabla f(z^\ell)-\nabla f(z^\star)+D^\top(\lambda^{\ell+1}-\lambda^\star)\right)\right\|_H\\ &\le \|z^\ell-z^\star\|_H+\|H^{-1/2}\|_2\cdot\|\nabla f(z^\ell)-\nabla f(z^\star)\|_2\\ &\quad+\left\|\Sigma^{-1/2}DH^{-1}D^\top\Sigma^{-1/2}\right\|_2\cdot\|\lambda^{\ell+1}-\lambda^\star\|_\Sigma.\end{aligned}\tag{5.13}$$

Applying the same reasoning as above, we define

$$\alpha_2=1+\left\|\Sigma^{-1/2}DH^{-1}D^\top\Sigma^{-1/2}\right\|_2(1+\alpha_1).\tag{5.14}$$

Then

$$\|\bar{z}^{\ell+1}-z^\star\|_H\le\alpha_2\cdot\|z^\ell-z^\star\|_H.\tag{5.15}$$

We can now bound the Lyapunov function as

$$\begin{aligned}\Omega(\bar{z}^{\ell+1},\lambda^{\ell+1}) &= \|\bar{z}^{\ell+1}-z^\star\|_H^2+\|\lambda^{\ell+1}-\lambda^\star\|_\Sigma^2\\ &\le \alpha_2^2\|z^\ell-z^\star\|_H^2+\alpha_1^2\|z^\ell-z^\star\|_H^2\\ &= \alpha^2\cdot\Phi(z^\ell-z^\star),\end{aligned}\tag{5.16}$$

where $\alpha^2=\alpha_1^2+\alpha_2^2$.

Finally, substituting this result into Lemma 6 gives

$$\Omega(\bar{z}^{\ell+1},\lambda^{\ell+1})+4\Phi(z^\ell-z^\star)\le\Omega(\bar{z}^\ell,\lambda^\ell)\tag{5.17a}$$

$$\left(1+\frac{4}{\alpha^2}\right)\Omega(\bar{z}^{\ell+1},\lambda^{\ell+1})\le\Omega(\bar{z}^\ell,\lambda^\ell),\tag{5.17b}$$

which implies the linear convergence bound

$$\Omega(\bar{z}^{\ell+1},\lambda^{\ell+1})\le\kappa\cdot\Omega(\bar{z}^\ell,\lambda^\ell),\tag{5.18}$$

where $\kappa=\frac{\alpha^2}{4+\alpha^2}<1$. This completes the proof. □

With the global linear convergence of Algorithm 2 established, we can use it to solve the distributed MPC problem and construct an effective real-time control algorithm.

5.1.1 Relaxed Recentered Logarithmic Barrier Functions

To enable an efficient real-time implementation of the MPC scheme, we propose a reformulation of the inequality constraints, which typically contribute to computational complexity, by embedding them into the objective function using relaxed recentered logarithmic barrier (RRLB) functions, as introduced in Feller and Ebenbauer (2018). This reformulation allows the controller to retain recursive feasibility guarantees. Furthermore, we can now ensure the closed-loop stability, based on analysis developed in Chapters 3 and 4, even in the presence of state constraints.

Alternative techniques in the literature, such as constraint tightening, have been widely used to ensure feasibility in the presence of suboptimality. However, these approaches tend to be overly conservative and can even exclude the optimal solution from the feasible set without yielding any reduction in computational complexity. In contrast, the use of RRLB barrier functions mitigates such conservatism and provides a more balanced trade-off between constraint satisfaction and computational efficiency.

Before introducing the RRLB function, it is important to define the structure of the MPC problem considered in this chapter. We focus on the predictive control of the LTI system described in (2.2), subject to both state and input constraints defined in (2.3). These constraints are assumed to satisfy the regularity conditions stated in Assumption 1 and are represented in the compact form given by (2.4).

Now, to recap the main idea of the RRLB function, we first define the relaxed barrier function $b_\delta : \mathbb{R} \to \mathbb{R}$ given by

$$b_\delta(y) = \begin{cases} -\ln(y), & \text{if } y > \delta, \\ \frac{1}{2}\left(\left(\frac{y-2\delta}{\delta}\right)^2 - 1\right) - \ln(\delta), & \text{if } y \leq \delta \end{cases} \tag{5.19}$$

where $\delta > 0$ is a relaxation parameter. The function b_δ is convex, globally twice continuously differentiable, and smoothly transitions into a quadratic penalty as y approaches or falls below δ. Fig. 5.1 visualizes how the choice of δ affects the value of the relaxed barrier function.

This function is used to define soft penalty terms for the inequality constraints (2.4)

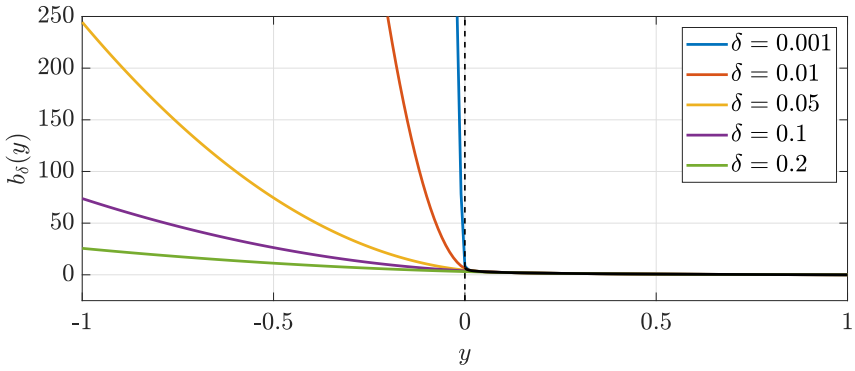


Figure 5.1: The graph illustrates the evolution of the recentered relaxed logarithmic barrier function for different values of the parameter δ.

in the MPC formulation. Specifically, we define

$$\begin{aligned}\widetilde{b}(x,u) = &\sum_{r=1}^{m_{\mathrm{x}}} w_r^{\mathrm{x}} \left(b_\delta([-G^{\mathrm{x}}x + d^{\mathrm{x}}]_r) + \ln([d^{\mathrm{x}}]_r)\right) \\ &+ \sum_{r=1}^{m_{\mathrm{u}}} w_r^{\mathrm{u}} \left(b_\delta([-G^{\mathrm{u}}u + d^{\mathrm{u}}]_r) + \ln([d^{\mathrm{u}}]_r)\right), \end{aligned} \tag{5.20}$$

where $w^{\mathrm{x}} \in \mathbb{R}_+^{m_{\mathrm{x}}}$ and $w^{\mathrm{u}} \in \mathbb{R}_+^{m_{\mathrm{u}}}$ are positive weights used to scale the barrier terms associated with each state and input constraint, respectively. Using the above penalty, we define the RRLB-based MPC problem

$$V(x(t)) = \min_{x,u} \ \mathcal{M}(x_N) + \sum_{k=0}^{N-1} L(x_k, u_k) \tag{5.21a}$$

$$\text{s.t.} \quad x_{k+1} = Ax_k + Bu_k, \quad k \in \{0, \dots, N-1\}, \tag{5.21b}$$

$$x_0 = x(t), \tag{5.21c}$$

resulting in an equality-constrained convex nonlinear problem. Here, the decision variables are stacked as $x \in \mathbb{R}^{(N+1)n_{\mathrm{x}}}$ and $u \in \mathbb{R}^{Nn_{\mathrm{u}}}$, and the augmented stage cost functions are defined as

$$L(x,u) = l(x,u) + \widetilde{b}(x,u), \ \mathcal{M}(x) = x^\top P x, \tag{5.22}$$

with $l(x,u) = x^\top Q x + u^\top R u$, where $Q, P \in \mathbb{S}_{++}^{n_{\mathrm{x}}}$, $R \in \mathbb{S}_{++}^{n_{\mathrm{u}}}$ being the standard weighting matrices for stage and terminal costs, respectively. The barrier weights w_{x} and w_{u} are chosen as described in Feller and Ebenbauer (2015) and further elaborated

in Feller and Ebenbauer (2018), such that the gradient of the stage cost satisfies $\nabla L(0,0) = 0$, which is essential for ensuring the stability of the resulting MPC controller.

Furthermore, the second derivatives of the barrier function $\widetilde{b}(x,u)$ are bounded (Feller and Ebenbauer, 2017) as follows

$$\nabla_x^2 \widetilde{b}(x,u) \preceq \Delta Q = \frac{1}{2\delta^2}(G^{\mathrm{x}})^\top \mathrm{diag}(\mathbf{1} + w^{\mathrm{x}}) G^{\mathrm{x}}, \tag{5.23a}$$

$$\nabla_u^2 \widetilde{b}(x,u) \preceq \Delta R = \frac{1}{2\delta^2}(G^{\mathrm{u}})^\top \mathrm{diag}(\mathbf{1} + w^{\mathrm{u}}) G^{\mathrm{u}}. \tag{5.23b}$$

As a consequence, the augmented stage cost $L(x,u)$ is bounded as

$$l(x,u) \leq L(x,u) \leq \widetilde{l}(x,u) = x^\top \widetilde{Q} x + u^\top \widetilde{R} u, \tag{5.24}$$

with $\widetilde{Q} = Q + \Delta Q$ and $\widetilde{R} = R + \Delta R$. The existence of these bounds is critical for the theoretical analysis of the proposed RRLB-based MPC.

Proposition 8. *Consider the linear system defined in* (2.2)*, where the pair* (A,B) *is controllable. Let the terminal cost function* $\mathcal{M}(\cdot)$ *be a Lyapunov function, i.e., it satisfies*

$$\mathcal{M}(A_K x) - \mathcal{M}(x) \leq L(x, Kx), \tag{5.25}$$

where K *is a pre-stabilizing state-feedback gain and* $A_K = A + BK$*. Then, the control input defined by recursively solving the RRLB-based MPC problem* (5.21) *asymptotically stabilizes the closed-loop system.*

This result ensures the nominal stability of the RRLB-based MPC formulation under standard assumptions. As shown in Feller and Ebenbauer (2017), the core idea is that if the terminal cost serves as a Lyapunov function for the system under a given feedback control law, then the associated MPC value function is also a Lyapunov function. This, in turn, guarantees the asymptotic stability of the closed-loop system. A formal proof of this result is available in Section IV.C of Feller and Ebenbauer (2017).

In the next section, we present a framework that combines the proposed RRLB-based MPC with the concept of distributed MPC, enabling efficient implementation in large-scale and structured systems.

5.2 Real-Time Distributed RRLB-Based MPC

In this section, we provide an overview of the newly proposed distributed RRLB-based MPC framework. We begin by identifying the class of systems for which this control policy is suitable. Next, we extend the centralized RRLB-based MPC approach from the previous section to a distributed setting, a contribution that has not been previously addressed in the literature. We also recall a procedure for designing a separable terminal cost that preserves nominal stability, enabling the distribution of RRLB-based MPC across the state-space domain. Finally, we formulate a real-time, distributed, and parallelizable RRLB-based MPC scheme using the ALADIN method to achieve an efficient control strategy.

5.2.1 Network-Structured Linear Systems

We consider a discrete-time LTI system structured as a network of M dynamically coupled subsystems. The global system dynamics is described by the model (2.2). Here we assume that the state matrix $A \in \mathbb{R}^{n_\mathrm{x} \times n_\mathrm{x}}$ is partitioned into blocks $A_{i,j} \in \mathbb{R}^{n_{\mathrm{x}_i} \times n_{\mathrm{x}_j}}$ as follows

$$A = \begin{pmatrix} A_{1,1} & \cdots & A_{1,M} \\ \vdots & \ddots & \vdots \\ A_{M,1} & \cdots & A_{M,M} \end{pmatrix}, \quad A_{i,j} = 0 \quad \text{if } (i,j) \notin \mathcal{E}, \tag{5.26}$$

and the input matrix is block-diagonal $B = \mathrm{diag}(B_1, \ldots, B_M)$, with $B_i \in \mathbb{R}^{n_{\mathrm{x}_i} \times n_{\mathrm{u}_i}}$. The states and control inputs

$$x_k = \left[x_{1,k}^\top, \ldots, x_{M,k}^\top\right]^\top, \quad u_k = \left[u_{1,k}^\top, \ldots, u_{M,k}^\top\right]^\top. \tag{5.27}$$

stack the local system states and control inputs of M subsystems. The system topology is described by the graph $(\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ is the index set of subsystems with $|\mathcal{N}| = M$, and $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ defines the set of interactions between subsystems in the form of the pairs. Specifically, if subsystem i has no influence on subsystem j, then $(i,j) \notin \mathcal{E}$. The structure of the matrix A reflects this topology, that is, $A_{i,j} = 0$ whenever $(i,j) \notin \mathcal{E}$.

Each subsystem $i \in \mathcal{N}$ has a local state vector $x_{i,k} \in \mathbb{R}^{n_{\mathrm{x}_i}}$ and local control input $u_{i,k} \in \mathbb{R}^{n_{\mathrm{u}_i}}$. The local update equation representing the dynamics of subsystem i is defined as

$$x_{i,k+1} = \sum_{j \in \mathcal{N}_i} A_{i,j} x_{j,k} + B_i u_{i,k} = A_{\mathcal{N}_i} x_{\mathcal{N}_i,k} + B_i u_{i,k}, \tag{5.28}$$

where $\mathcal{N}_i \subseteq \mathcal{N}$ denotes the set of neighboring subsystems that influence i, including i itself. The matrix $A_{\mathcal{N}_i}$ is constructed by collecting the blocks $A_{i,j} \in \mathbb{R}^{n_{\mathrm{x}_i} \times n_{\mathrm{x}_j}}$

corresponding to all $j \in \mathcal{N}_i$, where $A_{i,j}$ represents the coupling matrix between subsystems i and j. The vector $x_{\mathcal{N}_i,k}$ consists of the stacked states of the i-th subsystem and its neighboring subsystems, all indexed by $\mathcal{N}_i$.

In this formulation, coupling occurs exclusively through the state variables, while each subsystem remains independent in its control inputs. This setting is typical in distributed control applications, where control inputs are applied locally, but coordination across subsystems is required due to coupling in the system dynamics. Nonetheless, the proposed approach can also be extended to handle coupling in the input matrix B.

Additionally, we assume that each subsystem is subject to local polyhedral state and input constraints. Specifically, the global constraint sets are defined as the Cartesian products

$$\mathbb{X} = \mathbb{X}_1 \times \cdots \times \mathbb{X}_M, \quad \mathbb{U} = \mathbb{U}_1 \times \cdots \times \mathbb{U}_M, \tag{5.29}$$

where $\mathbb{X}_i \subset \mathbb{R}^{n_{\mathrm{x}_i}}$ and $\mathbb{U}_i \subset \mathbb{R}^{n_{\mathrm{u}_i}}$ represent the admissible sets for subsystem i.

5.2.2 Design of Separable Terminal Cost

To guarantee closed-loop stability of the nominal RRLB-based MPC controller, it is essential to design a separable terminal cost that serves as a Lyapunov function, as highlighted in Proposition 8. To enable this in a distributed setting, we begin by recalling the following structural assumption from Conte et al. (2016).

Assumption 9. *There exists a linear feedback control law* $\mu(x) = Kx = \mathrm{col}_{i\in\mathcal{N}}(K_{\mathcal{N}_i} x_{\mathcal{N}_i})$ *such that the LTI system* (2.2) *is asymptotically stable under* $u = \mu(x)$.

Here, the $\mathrm{col}_{i\in\mathcal{N}}(K_{\mathcal{N}_i} x_{\mathcal{N}_i})$ denotes the vector consisting of the stacked subvectors $K_{\mathcal{N}_i} x_{\mathcal{N}_i}$ for all $i \in \mathcal{N}$, the vector $x_{\mathcal{N}_i}$ aggregates the local states x_j for all $j \in \mathcal{N}_i$, and the matrix K is block-structured to reflect the network topology. Specifically, block $K_{i,j} \neq 0$ if $j \in \mathcal{N}_i$, and $K_{i,j} = 0$ otherwise. The matrix $K_{\mathcal{N}_i}$ then collects the nonzero feedback gains relevant to subsystem i. For compactness, we define selection matrices $T_i \in \{0,1\}^{n_{\mathrm{x}_i} \times n_{\mathrm{x}}}$ and $W_i \in \{0,1\}^{n_{\mathrm{x}_{\mathcal{N}_i}} \times n_{\mathrm{x}}}$ such that $x_i = T_i x$ and $x_{\mathcal{N}_i} = W_i x$.

Proposition 9 (Theorem 6 in Conte et al. (2016))**.** *Let Assumption 9 hold. If there exist functions* $\vartheta_i(\cdot)$ *such that*

$$\mathcal{M}_i(x_i^+) - \mathcal{M}_i(x_i) \leq -\widetilde{l}_i(x_i, K_{\mathcal{N}_i} x_{\mathcal{N}_i}) + \vartheta_i(x_{\mathcal{N}_i}), \tag{5.30a}$$

$$\sum_{i=1}^{M} \vartheta_i(x_{\mathcal{N}_i}) \leq 0, \tag{5.30b}$$

with $x_i^+ = A_{\mathcal{N}_i} x_{\mathcal{N}_i} + B_i u_i$*, then the overall terminal cost*

$$\mathcal{M}(x) = \sum_{i=1}^{M} \mathcal{M}_i(x_i) \tag{5.31}$$

is a Lyapunov function for system (2.2) *under the feedback law* $u = \mu(x)$*.*

We now tailor design procedure, introduced in Conte et al. (2016), to obtain a separable terminal cost $\mathcal{M}(x) = \sum_i \mathcal{M}_i(x_i)$ that satisfies (5.30). Let $\vartheta_i(x_{\mathcal{N}_i}) = x_{\mathcal{N}_i}^\top \Gamma_{\mathcal{N}_i} x_{\mathcal{N}_i}$ be a nonnegative quadratic function. Then the matrices P_i, $\Gamma_{\mathcal{N}_i}$, and feedback gains $K_{\mathcal{N}_i}$ can be determined by solving the following Linear Matrix Inequality (LMI) problem

$$\min_{E_i, F_{\mathcal{N}_i}, E_{\mathcal{N}_i}, Y_{\mathcal{N}_i}} \; -\sum_{i=1}^{M} \log\det(E_i) \tag{5.32a}$$

subject to $\forall\, i \in \mathcal{N}$,

$$\begin{bmatrix} \overline{E}_i + F_{\mathcal{N}_i} & E_{\mathcal{N}_i} A_{\mathcal{N}_i}^\top + Y_{\mathcal{N}_i}^\top B_i^\top & E_{\mathcal{N}_i} \mathcal{Q}_i^{1/2} & Y_{\mathcal{N}_i}^\top R_i^{1/2} \\ A_{\mathcal{N}_i} E_{\mathcal{N}_i} + B_i Y_{\mathcal{N}_i} & E_i & 0 & 0 \\ \mathcal{Q}_i^{1/2} E_{\mathcal{N}_i} & 0 & I & 0 \\ R_i^{1/2} Y_{\mathcal{N}_i} & 0 & 0 & I \end{bmatrix} \succeq 0, \tag{5.32b}$$

$$\sum_{i=1}^{M} W_i^\top F_{\mathcal{N}_i} W_i \preceq 0, \tag{5.32c}$$

where $\mathcal{Q}_i = \partial^2_{x_{\mathcal{N}_i}} \tilde{l}_i(x_i, u_i)$, $E_i = P_i^{-1}$, $\overline{E}_i = W_i T_i^\top E_i T_i W_i^\top$, $E_{\mathcal{N}_i} = W_i E W_i^\top$, $F_{\mathcal{N}_i} = E_{\mathcal{N}_i} \Gamma_{\mathcal{N}_i} E_{\mathcal{N}_i}$, and $Y_{\mathcal{N}_i} = K_{\mathcal{N}_i} E_{\mathcal{N}_i}$. Feasibility of this LMI guarantees that the conditions in (5.30) are satisfied. Note that this offline procedure does not affect the real-time runtime of the MPC controller.

Next, we extend the nominal closed-loop stability result for RRLB-based MPC from Feller and Ebenbauer (2017) to the distributed setting with a separable terminal cost, as formalized in the following theorem.

Theorem 10. *Let Assumptions 8 and 9 hold, and let the LMI problem* (5.32) *be feasible. Then, for any feasible initial state* x_0*, the control action generated as the solution of the RRLB-based MPC problem* (5.21) *with separable stage and terminal costs*

$$L(x, u) = \sum_{i=1}^{M} L_i(x_i, u_i), \quad \mathcal{M}(x) = \sum_{i=1}^{M} \mathcal{M}_i(x_i),$$

asymptotically stabilizes the LTI system (2.2)*.*

Proof. The result follows directly from Propositions 8 and 9. The separable terminal cost $\mathcal{M}(x)$ constructed via the LMI (5.32) satisfies the Lyapunov conditions (5.30), which in turn ensures that the value function $V(x)$ decreases along closed-loop trajectories. Specifically, we have

$$
\begin{aligned}
V(x_1^\star) - V(x_0) + \sum_{i=1}^{M} L_i(x_{i,0}, u_{i,0}^\star) & \\
&\leq \sum_{i=1}^{M} \Big(\mathcal{M}_i(x_{i,N}^+) - \mathcal{M}_i(x_{i,N}^\star) + L_i(x_{i,N}^\star, K_{\mathcal{N}_i} x_{\mathcal{N}_i,N}^\star) \Big) && \text{(5.33a)} \\
&\overset{(5.24)}{\leq} \sum_{i=1}^{M} \Big(\mathcal{M}_i(x_{i,N}^+) - \mathcal{M}_i(x_{i,N}^\star) + \widetilde{l}_i(x_{i,N}^\star, K_{\mathcal{N}_i} x_{\mathcal{N}_i,N}^\star) \Big) && \text{(5.33b)} \\
&\overset{(5.30a)}{\leq} \sum_{i=1}^{M} \vartheta_i(x_{\mathcal{N}_i,N}^\star) \overset{(5.30b)}{\leq} 0 && \text{(5.33c)}
\end{aligned}
$$

where $x_1^\star = Ax_0 + Bu_0^\star$, representing state evolution under the optimal control action, and $x_{i,N}^+ = (A_{\mathcal{N}_i} + BK_{\mathcal{N}_i}) x_{\mathcal{N}_i,N}^\star$, denoting state evolution under any stabilizing feedback control law. This guarantees that $V(x)$ is a Lyapunov function, proving the asymptotic stability of the closed-loop system. □

Note that this nominal stability result assumes the MPC problem (5.21) is solved to optimality. In practical implementations, computational limitations or strict sampling intervals may require early termination of the optimization process, which is addressed by the fixed-iteration scheme proposed in the subsequent sections.

5.2.3 Distributed RRLB-Based MPC Algorithm

This section introduces a real-time distributed RRLB-based MPC algorithm. The objective is to approximate the solution of Problem (5.21) in a distributed manner while preserving the closed-loop asymptotic stability guarantees established in the nominal case. To facilitate the algorithmic design and notation, we introduce a condensed representation of the system dynamics and cost.

The local decision variables for subsystem i at step k of the prediction horizon are defined as

$$
z_{i,0} = u_{i,0}, \quad z_{i,k} = \left[x_{i,k}^\top, u_{i,k}^\top \right]^\top \text{ for } k \in \{1, \ldots, N-1\}, \quad z_{i,N} = x_{i,N}.
$$

Each local cost is denoted by $f_{i,k}(z_{i,k}) = L_i(x_{i,k}, u_{i,k})$ for $k \in \{0, \ldots, N-1\}$, and $f_{i,N}(z_{i,N}) = \mathcal{M}_i(x_{i,N})$ for the terminal stage.

To express the subsystem dynamics in a compact form, we reformulate (5.28) as

$$\mathcal{C}_{i,k} z_{i,k+1} + \sum_{j \in \mathcal{N}_i} \mathcal{A}_{i,j,k} z_{j,k} = h_{i,k}, \tag{5.34}$$

where the matrices $\mathcal{A}_{i,j,k}$, $\mathcal{C}_{i,k}$, and the vector $h_{i,k}$ are defined as follows. At the initial time step $k = 0$, the coefficients are given by $\mathcal{A}_{i,i,0} = B_i$ and $\mathcal{A}_{i,j,0} = 0$ for all $j \in \mathcal{N}_i \setminus \{i\}$, while the corresponding right-hand side reads $h_{i,0} = -A_{\mathcal{N}_i} x_{\mathcal{N}_i,0}$. For all subsequent time steps $k \geq 1$, the system matrices are specified by $\mathcal{A}_{i,i,k} = [A_{i,i},\ B_i]$ and $\mathcal{A}_{i,j,k} = [A_{i,j},\ 0]$ for $j \in \mathcal{N}_i \setminus \{i\}$, with $h_{i,k} = 0$. In all instances, the matrix $\mathcal{C}_{i,k} = [-I,\ 0]$ acts as a selector that extracts the state component from the decision variable $z_{i,k}$.

Using this compact representation, the distributed MPC problem can be formulated as

$$V(x_0) = \min_{z} \sum_{i=1}^{M} \sum_{k=0}^{N} f_{i,k}(z_{i,k}) \tag{5.35a}$$

$$\text{s.t. } \mathcal{C}_{i,k} z_{i,k+1} + \sum_{j \in \mathcal{N}_i} \mathcal{A}_{i,j,k} z_{j,k} = h_{i,k}, \quad \forall i \in \mathcal{N}, \quad k = \{0, \ldots, N-1\}. \tag{5.35b}$$

The Lagrangian function associated with (5.35) is defined as

$$\begin{aligned} \mathcal{L}(z, \lambda) = &\sum_{i=1}^{M} \left(\mathcal{L}_{i,N}(z_{i,N}, \lambda_{i,N-1}) + \mathcal{L}_{i,0}(z_{i,0}, \lambda_{\mathcal{N}_i,0}) \right) \\ &+ \sum_{i=1}^{M} \sum_{k=1}^{N-1} \mathcal{L}_{i,k}(z_{i,k}, \lambda_{\mathcal{N}_i,k}, \lambda_{i,k-1}), \end{aligned} \tag{5.36}$$

where each $\lambda_{\mathcal{N}_i,k}$ stacks dual variables associated with subsystem i's neighborhood at stage k. The Lagrangian function (5.36) can be decomposed into the local Lagrangian functions as

$$\mathcal{L}_{i,0}(z_{i,0}, \lambda_{\mathcal{N}_i,0}) = f_{i,0}(z_{i,0}) + z_{i,0}^\top \left(\sum_{j \in \mathcal{N}_i} \mathcal{A}_{j,i,k}^\top \lambda_{j,0} \right) + h_{i,0}^\top \lambda_{i,0}, \tag{5.37}$$

$$\mathcal{L}_{i,k}(z_{i,k}, \lambda_{\mathcal{N}_i,k}, \lambda_{i,k-1}) = f_{i,k}(z_{i,k}) + \lambda_{i,k-1}^\top \mathcal{C}_{i,k-1} z_{i,k} + z_{i,k}^\top \left(\sum_{j \in \mathcal{N}_i} \mathcal{A}_{j,i,k}^\top \lambda_{j,k} \right) \tag{5.38}$$

$$\mathcal{L}_{i,N}(z_{i,N}, \lambda_{i,N-1}) = f_{i,N}(z_{i,N}) + \lambda_{i,N-1}^\top \mathcal{C}_{i,N-1} z_{i,N}, \tag{5.39}$$

Now, to simplify the notation, we define the penalty function as

$$\Psi(z) = \sum_{i=1}^{M} \sum_{k=0}^{N} \Psi_{i,k}(z_{i,k}), \tag{5.40}$$

with stage-wise components, $\Psi_{i,0}(z_{i,0}) = z_{i,0}^{\top} R_i z_{i,0}$, $\Psi_{i,k}(z_{i,k}) = z_{i,k}^{\top} H_i z_{i,k}$, $\Psi_{i,N}(z_{i,N}) = \mathcal{M}_i(z_{i,N})$, and $H_i = \mathrm{diag}(Q_i, R_i)$.

Algorithm 5 outlines the procedure for solving the MPC problem (5.35). It begins with the initialization of the global variables $(\bar{z}^1, \lambda^1)$, which serve as initial guesses for the optimization process. In Step 1), these initial guesses are rescaled to ensure closed-loop stability. After rescaling, the ALADIN method is applied with a fixed number of iterations to solve (5.35) in a distributed manner. Step 2a) involves solving decoupled convex NLPs, preferably in parallel, to compute local solutions. These local solutions are then used in Step 2b), where a coupled QP is solved to obtain globally consistent variables, which serve as references for the next iteration of local NLPs. Once the maximum number of iterations $\bar{\ell}$ is reached, the ALADIN method terminates, and the resulting control input is applied to the system (2.2) in Step 3). Finally, the initial guesses are updated via forward shifting in Step 4), and the entire procedure is repeated at the next control step. Note that full decentralization, i.e., the elimination of the central coordination unit, in Algorithm 5 is possible, as discussed in Section 2.2.2.

As discussed, Algorithm 5 employs a fixed-iteration strategy that eliminates the runtime variability typically associated with iterative solvers using an adaptive tolerance-based stopping criterion. While Algorithm 5 appears similar to Algorithm 4 from Chapter 4, it differs in several key aspects: it does not rely on explicit MPC, it employs RRLB functions to handle constraints, and it introduces an additional layer of distribution across the state space. Within this framework, dualizing the initial condition offers no computational or memory advantage and is therefore omitted.

By combining spatial and temporal decomposition, i.e., both through state-space and prediction horizon, the controller solves multiple small-scale unconstrained NLPs in parallel, followed by a single structured QP, schematically illustrated in Fig. 5.2. This design is both computationally efficient and well-suited for real-time deployment on embedded control platforms. Furthermore, it scales effectively to large-scale systems with high-dimensional state space, which can be decomposed into simpler, low-dimensional subproblems.

Algorithm 5 Distributed real-time iteration for cooperative MPC.

Initialization: choose a initial guess $(\bar{z}^1, \lambda^1)$, a constant $\gamma > 0$, and a maximum number $\bar{\ell}$ of iterations.

Online:

1) Obtain state measurement $x(t)$ and evaluate $\Gamma^1 = \|\bar{z}^1\|_H^2 + \|\lambda^1\|_\Sigma^2$. If $\Gamma^1 > \gamma^2 \|x(t)\|_Q^2$, rescale initial guesses

$$\bar{z}^1 = \bar{z}^1 \sqrt{\frac{\gamma^2 \|x(t)\|_Q^2}{\Gamma^1}}, \quad \lambda^1 = \lambda^1 \sqrt{\frac{\gamma^2 \|x(t)\|_Q^2}{\Gamma^1}}.$$

2) **For:** $\ell \to \bar{\ell}$

   (a) Solve decoupled convex NLPs in parallel

$$\min_{z_{i,0}} \mathcal{L}_{i,0}(z_{i,0}, \lambda^\ell_{\mathcal{N}_i,0}) + \Psi_{i,0}(z_{i,0} - \bar{z}^\ell_{i,0}), \tag{5.41a}$$

$$\min_{z_{i,k}} \mathcal{L}_{i,k}(z_{i,k}, \lambda^\ell_{\mathcal{N}_i,k}, \lambda^\ell_{i,k-1}) + \Psi_{i,k}(z_{i,k} - \bar{z}^\ell_{i,k}), \tag{5.41b}$$

$$\min_{z_{i,N}} \mathcal{L}_{i,N}(z_{i,N}, \lambda^\ell_{i,N-1}) + \Psi_{i,N}(z_{i,N} - \bar{z}^\ell_{i,N}), \tag{5.41c}$$

   to achieve solutions $z^\ell_{i,k}$ for all $i \in \mathcal{N}$ and $k \in \{1, \dots, N-1\}$.

   (b) Solve coupled QP problem

$$\begin{aligned} \min_{\tilde{z}} \ & \sum_{i=1}^{M} \sum_{k=0}^{N} \frac{1}{2} \Psi_{i,k}(\bar{z}_{i,k} - 2z^\ell_{i,k} + \bar{z}^\ell_{i,k}) \\ \text{s.t. } & \mathcal{C}_{i,k} \bar{z}_{i,k+1} + \sum_{j \in \mathcal{N}_i} \mathcal{A}_{i,j,k} \bar{z}_{j,k} = h_{i,k}, \qquad \mid \Delta\lambda_{i,k} \end{aligned} \tag{5.42}$$

   $k \in \{0, \dots, N-1\}$, $i \in \mathcal{N}$, update $\bar{z}^{\ell+1} = \bar{z}$, $\quad \lambda^{\ell+1} = \lambda^\ell + \Delta\lambda^\ell$.

3) Apply $u_0 = \mathrm{col}_{i \in \mathcal{N}}(z^{\bar{\ell}}_{i,0})$ to the system in (2.2).

4) Update initial guesses for each $i \in \mathcal{N}$ $\bar{z}^1_i = [z^{\bar{\ell}\top}_{i,1}, \dots, z^{\bar{\ell}\top}_{i,N}, z^{\bar{\ell}\top}_{i,N}]^\top$, $\lambda^1_i = [\lambda^{\bar{\ell}\top}_{i,1}, \dots, \lambda^{\bar{\ell}\top}_{i,N-1}, \lambda^{\bar{\ell}\top}_{i,N-1}]^\top$ and go to Step 1.

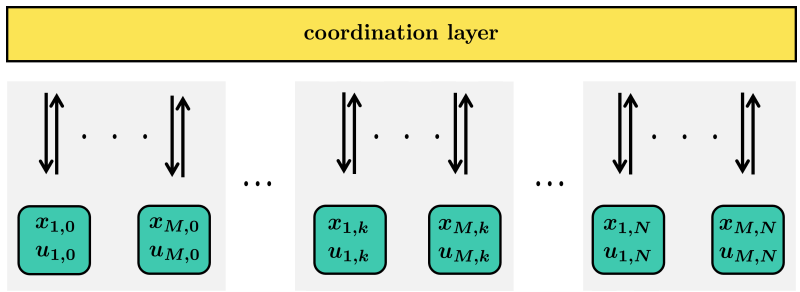


Figure 5.2: Visualization of the temporal and spatial decomposition of the MPC problem resulting in small-scale subproblems.

5.2.3.1 Closed-Loop Stability of Algorithm 5

In this section, we analyze the closed-loop stability properties of the proposed real-time distributed MPC procedure described in Algorithm 5. Recall that the algorithm exhibits a global linear convergence rate $\kappa < 1$, as established earlier. To guarantee closed-loop asymptotic stability under suboptimal control law, resulting from the fixed-iteration procedure in Algorithm 5, we need the following result.

Lemma 11. *Let $\mathcal{V}_\varrho = \{x \in \mathbb{R}^{n_x} : V(x) \leq \varrho\}$ denote a sublevel set of the value function $V(x)$ associated with Problem (5.21), for a given $\varrho > 0$. Then, there exist constants $\eta_1, \eta_2 > 0$ such that for any $x(t_1), x(t_2) \in \mathcal{V}_\varrho$, the following inequality holds*

$$|V(x(t_1)) - V(x(t_2))| \leq \eta_1 \|x(t_1) - x(t_2)\|_Q + \eta_2 \|x(t_1) - x(t_2)\|_Q^2. \tag{5.43}$$

Proof. The RRLB-based MPC problem (5.21) is a parametric nonlinear program where the initial state x_0 serves as a parameter. By applying a single shooting, the dynamic constraints can be eliminated, resulting in a parametric objective of the form

$$V(x_0) = f(z^\star(x_0), x_0), \tag{5.44}$$

where $z^\star(x_0)$ denotes the optimal solution as a function of the initial state.

Then, from the first-order optimality condition, we have

$$\nabla_y f(z^\star(x_0), x_0) = 0 \quad \Rightarrow \quad \nabla_{zz} f(z^\star(x_0), x_0) \frac{\mathrm{d}z^\star}{\mathrm{d}x_0} + \nabla_{zx_0} f(z^\star(x_0), x_0) = 0, \tag{5.45}$$

that holds for any $x_0 \in V_\varrho$. Given that f is twice continuously differentiable, strongly convex, and its Hessian is lower and upper bounded, the solution map $z^\star(\cdot)$ is Lipschitz

continuously differentiable. Thus, there exists constant $L > 0$ such that

$$\|z^\star(x(t_1)) - z^\star(x(t_2))\| \leq L\|x(t_1) - x(t_2)\|. \tag{5.46}$$

holds for any $x_1(t)$, $x_2(t) \in V_\varrho$. For detailed proof, see Bonnans and Shapiro (2013). Using the strong convexity of f, we have

$$\begin{aligned} V(x(t_1)) - V(x(t_2)) &= f(z^\star(x(t_1)), x(t_1)) - f(z^\star(x(t_2)), x(t_2)) && \text{(5.47a)} \\ &\leq -\nabla_z f(z^\star(x(t_1)), x(t_1)) - \underline{\eta}\|z^\star(x(t_1)) - z^\star(x(t_2))\|_2^2 && \text{(5.47b)} \end{aligned}$$

that holds for a constant $\underline{\eta} > 0$. Moreover, using the Lipschitz continuity of ∇f, we have

$$\begin{aligned} V(x(t_1)) - V(x(t_2)) &= f(z^\star(x(t_1)), x(t_1)) - f(z^\star(x(t_2)), x(t_2)) && \text{(5.48a)} \\ &\leq \nabla_z f(z^\star(x(t_2)), x(t_2)) + \bar{\eta}\|z^\star(x(t_1)) - z^\star(x(t_2))\|_2^2 && \text{(5.48b)} \end{aligned}$$

that is satisfied for a constant $\bar{\eta} > 0$. Combining these two inequalities leads to

$$\begin{aligned} &|V(x(t_1)) - V(x(t_2))| \\ &\quad \leq \|\nabla_z f(z^\star(x(t_1)), x(t_1)) - \nabla_z f(z^\star(x(t_2)), x(t_2))\|_2 \\ &\qquad + |\bar{\eta} - \underline{\eta}| \cdot \|z^\star(x(t_1)) - z^\star(x(t_2))\|_2^2 && \text{(5.49a)} \\ &\quad \leq c \cdot \|z^\star(x(t_1)) - z^\star(x(t_2))\|_2 + |\bar{\eta} - \underline{\eta}| \cdot \|z^\star(x(t_1)) - z^\star(x(t_2))\|_2^2 && \text{(5.49b)} \\ &\quad \leq \eta_1\|x(t_1) - x(t_2)\|_2 + \eta_2\|x(t_1) - x(t_2)\|_2^2, && \text{(5.49c)} \end{aligned}$$

where the second step holds because of the Lipschitz continuity of ∇f, and the final step uses (5.46). □

With inequality (5.43) established, the final step toward proving closed-loop asymptotic stability follows from the value function descent property and convergence result stated in Theorem 5 of Chapter 4. When the fixed number of iterations $\bar{\ell} \in \mathbb{N}$ in Algorithm 5 is chosen sufficiently large, the suboptimal control input $u_0^{\bar{\ell}}$ guarantees stability of the closed-loop system.

Specifically, asymptotic stability is achieved if the number of iterations satisfies

$$\bar{\ell} > \frac{2\log\left(2\eta_1\gamma\sqrt{\frac{\sigma(1+\kappa)}{\kappa}} + 2\eta_2\gamma^2\left(\frac{1+\kappa}{\kappa}\right)\right)}{\log(1/\kappa)}, \tag{5.50}$$

where σ, κ, η_1, and η_2 are constants as defined above and in previous results. This condition ensures that the value function decreases at every control step, thereby establishing the asymptotic stability of the closed-loop system under the proposed real-time distributed MPC scheme, with constraints incorporated via RRLB functions.

5.2.3.2 Implementation Details

This section discusses the numerical implementation aspects of the proposed distributed RRLB-based MPC procedure presented in Algorithm 5. The implementation is tailored to ensure computational efficiency and suitability for real-time applications, particularly when deployed on embedded or industrial hardware.

In Step 2a) of Algorithm 5, a sequence of unconstrained convex NLPs is solved in parallel. To solve these subproblems efficiently, a Newton-type optimization method is employed. The Newton iteration m for subsystem i at step k is given by

$$\psi_{i,k}^{m+1} = \psi_{i,k}^{m} - \left(\nabla^2_{\psi_{i,k}} J_{i,k}(\psi_{i,k}^{m})\right)^{-1} \nabla_{\psi_{i,k}} J_{i,k}(\psi_{i,k}^{m}), \tag{5.51}$$

where $J_{i,k}(\cdot)$ denotes the objective function of the decoupled NLPs for step $k \in \{0, \dots, N\}$, defined as

$$J_{i,0}(z_{i,0}) = \mathcal{L}_{i,0}(z_{i,0}, \lambda^{\ell}_{\mathcal{N}_i,0}) + \Psi_{i,0}(z_{i,0} - \bar{z}^{\ell}_{i,0}), \tag{5.52a}$$

$$J_{i,k}(z_{i,k}) = \mathcal{L}_{i,k}(z_{i,k}, \lambda^{\ell}_{\mathcal{N}_i,k}, \lambda^{\ell}_{i,k-1}) + \Psi_{i,k}(z_{i,k} - \bar{z}^{\ell}_{i,k}), \ k \in \{1, \dots, N-1\}, \tag{5.52b}$$

$$J_{i,N}(z_{i,N}) = \mathcal{L}_{i,N}(z_{i,N}, \lambda^{\ell}_{i,N-1}) + \Psi_{i,N}(z_{i,N} - \bar{z}^{\ell}_{i,N}), \tag{5.52c}$$

for each $i \in \mathcal{N}$ in iteration ℓ of Algorithm 5 and with fixed values of global results λ^{ℓ} and $\bar{z}^{\ell}$. Each local problem is solved using a fixed number of Newton steps $m \to \bar{m}$, as commonly practiced in real-time optimization (Boyd and Vandenberghe, 2004; Feller and Ebenbauer, 2018). While the theoretical convergence guarantees in Theorem 7 assume optimal NLP solutions, in practice, early termination after $\bar{m}$ iterations is sufficient to maintain good closed-loop performance. Therefore, the local solution $z_{i,k} = \psi_{i,k}^{\bar{m}}$ serves as input to Step 2b). Furthermore, the inclusion of spatial decomposition reduces the dimension of the local Hessian matrices, making matrix inversion more computationally efficient and well-suited to parallel computing architectures.

In Step 2b), the algorithm solves the coupled equality-constrained QP (5.42), which exhibits a structure analogous to a LQR problem. If a central coordinator with access to global system information is available, the problem can be solved efficiently using the Riccati recursion. Specifically, the recursion is performed backward in time as

$$\mathcal{P}_{k-1} = Q + A^{\top}\mathcal{P}_k A - (A^{\top}\mathcal{P}_k B)(B^{\top}\mathcal{P}_k B + R)^{-1}(B^{\top}\mathcal{P}_k A), \tag{5.53}$$

with terminal term $\mathcal{P}_N = \operatorname{diag}_{i\in\mathcal{N}}(P_i)$. The corresponding control gains are computed as

$$K_k = -(B^{\top}\mathcal{P}_k B + R)^{-1} B^{\top}\mathcal{P}_k A. \tag{5.54}$$

These gains are used to analytically compute the optimal global update $\bar{z}^{\ell+1}$ and the associated dual update $\Delta\lambda^{\ell}$.

We recall that in settings where only local communication among neighboring subsystems is allowed, solving the coupled QP centrally may not be feasible. In such cases, iterative decentralized methods, such as the conjugate gradient method, can be employed to solve (5.42) in a fully distributed fashion. For more details on these techniques, see Engelmann et al. (2020).

Importantly, the entire procedure defined in Algorithm 5 is fully self-contained and does not require the use of external general-purpose solvers. This enables reliable and efficient implementation in environments with strict memory or computational constraints.

5.3 Case Study

To demonstrate the benefits of the proposed distributed RRLB-based MPC method, we used the spring-vehicle-damper system with coupled inverted pendulums, shown in Figure 5.3.

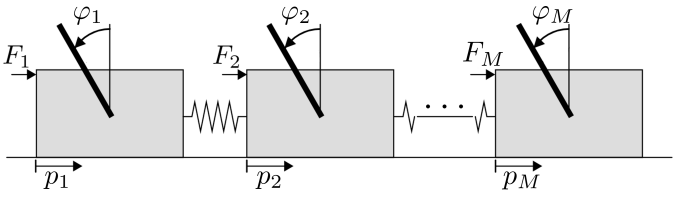


Figure 5.3: Unstable spring-vehicle damper system with inverted pendulums.

The system's connectivity graph is defined as $\mathcal{N}_i = \{i-1, i+1\}$ for $i \in \{2, 3, \ldots, M-1\}$, with $\mathcal{N}_1 = \{2\}$ and $\mathcal{N}_M = \{M-1\}$ for the first and last trolleys. The state vector $x_{i,k} = [p_{i,k}, \dot{p}_{i,k}, \varphi_{i,k}, \dot{\varphi}_{i,k}]^\top$ in step k, includes the position and velocity of the cart, and the angle and angular velocity of the pendulum. The control inputs are the forces applied to the respective cart $u_{i,k} = F^{\mathrm{c}}_{i,k}$. The continues-time equations for the

pendulum dynamics are defined as follows

$$\ddot{p}_i = \frac{F_i^{\mathrm{c}} + \frac{3}{4} m \cdot g \sin(\varphi_i)\cos(\varphi_i) - \frac{m \cdot l}{2}\dot{\varphi}_i^2 \sin(\varphi_i) + F_i^{\mathrm{left}} + F_i^{\mathrm{right}}}{M_{\mathrm{c}} + m - \frac{3}{4}(\cos(\varphi_i))^2} \tag{5.55a}$$

$$\ddot{\varphi} = \frac{3g}{2l}\sin(\varphi_i) + \frac{3}{2l}\cos(\varphi_i)\ddot{p}_i, \tag{5.55b}$$

with spring stiffness $k = 0.1\,\mathrm{N/m}$, cart masses $M_{\mathrm{c}} = 2\,\mathrm{kg}$, pendulum masses $m = 0.25\,\mathrm{kg}$, length of each pendulum $l = 0.2\,\mathrm{m}$, gravitational constant $g = 9.81\,\mathrm{m/s^2}$ and coupling forces between neighboring carts (where applicable) as $F_i^{\mathrm{left}} = k(q_{i-1} - q_i)$ and $F_i^{\mathrm{right}} = k(q_{i+1} - q_i)$. Since our framework is designed for LTI systems, we adopted a linearization approach similar to that used in Stomberg et al. (2024), where the nonlinear model was first discretized using a fourth-order explicit Runge-Kutta method with a sampling time of 40 ms, and then linearized.

Simulations were executed using `MATLAB R2024a` on a computer equipped with an AMD Ryzen 7 PRO CPU (8 cores) and 16 GB of RAM. The objective was to evaluate the performance of the proposed distributed MPC scheme, as implemented in Algorithm 5, in comparison with conventional MPC approaches.

The controller was designed with weighting matrices $Q_i = \mathrm{diag}(1,\, 0.1,\, 10,\, 0.1)$, $R_i = 10^{-4}$, and terminal weighting matrix P_i computed by solving the LMI condition (5.32) for all $i \in \mathcal{N}$. Furthermore, we used a sampling period of $T_{\mathrm{s}} = 0.04\,\mathrm{s}$, the same as for the discretization. The velocity and input constraints were specified as $-5 \leq \dot{p}_{i,k} \leq 5$ and $-100 \leq u_{i,k} \leq 100$, respectively. Each subsystem was initialized with $x_{i,0} = [i,\ 0,\ \pi,\ 0]^\top$, and the regularization parameter for the RRLB formulation was set to $\delta = 10^{-4}$.

Each decoupled subproblem in Algorithm 5 was solved using the Newton method with a cap of two iterations per control step, as it has been observed that this is sufficient to reach the tolerance 10^{-6}. Figure 5.4 illustrates the evolution of the closed-loop cost

$$J_{\mathrm{CL}}(t) = x(t)^\top Q x(t) + u(t)^\top R u(t)$$

over time for a system of $M = 8$ interconnected trolleys, with a prediction horizon of $N = 40$, and different values of the maximum iteration count $\bar{\ell}$. As illustrated, the distributed controller achieves a more accurate solution with higher iteration counts. However, the accuracy achieved with $\bar{\ell} = 2$ is already sufficient for real-time implementation. It is also worth noting that the initial guesses in Algorithm 5 were set to zero vectors, suggesting that even better accuracy could be achieved through warm-starting techniques.

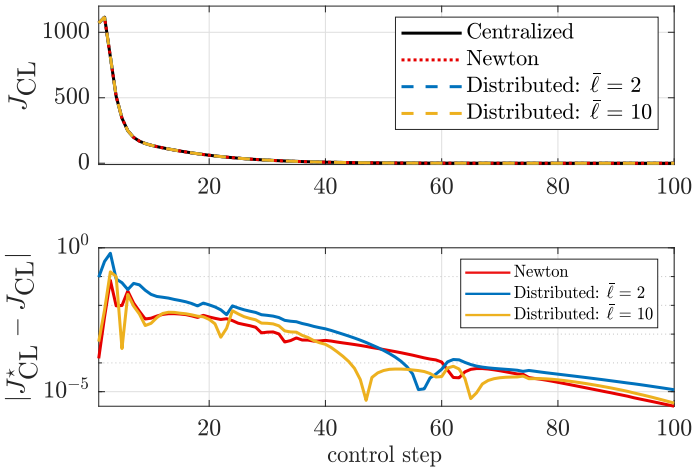


Figure 5.4: Comparison of closed-loop performance between Algorithm 5 with varying values of $\bar{\ell}$ and the Newton method, evaluated against the nominal MPC reference.

Furthermore, the state and input trajectories are shown in Fig. 5.5 and Fig. 5.6, this time for a system with $M = 3$ interconnected carts to improve plot clarity and readability. The prediction horizon remains unchanged $N = 40$. As shown, the input constraints are active during the first two time steps, while the state constraints are active between time steps 6 and 23. No constraint violations were observed. As in the previous case, the controller based on Algorithm 5 with $\bar{\ell} = 2$ again achieves near-optimal control performance.

Table 5.1 presents a comparison of average t_{avg} and maximum $t_{\max}$ runtime performance in milliseconds for three solvers: `GUROBI`, `OSQP`, and our implementation of Algorithm 5 with $\bar{\ell} = 2$. While `OSQP` achieves lower runtimes, it occasionally violates constraints, leading to infeasibility in open-loop simulations. In contrast, our method ensures feasibility and yields performance comparable to centralized solvers. It is worth noting that our prototype does not yet include parallel evaluation of the decoupled subproblems, which presents further potential for runtime optimization.

For context, `quadprog` was also tested and found to be approximately twice as slow as `GUROBI`. Additionally, a standard Newton-based method exhibited runtimes up to ten times longer due to the complexity of the benchmark problem.

Table 5.1: Runtime analysis (in milliseconds) of Algorithm 5 compared to the `GUROBI` and `OSQP` solvers.

M	N	`GUROBI`		`OSQP`		Algorithm 5 with $\bar{\ell} = 2$	
		t_{avg}	$t_{\max}$	t_{avg}	$t_{\max}$	t_{avg}	$t_{\max}$
5	40	20.2	26.6	7.4	61.2	18.1	23.3
5	60	30.6	35.4	11.8	94.5	26.0	30.0
5	80	78.3	91.6	15.6	117.9	33.8	38.0
8	40	80.9	105.9	27.5	139.9	26.0	34.8
8	60	100.9	139.6	45.2	230.3	37.6	58.5
8	80	122.4	148.2	85.7	484.5	49.4	59.3

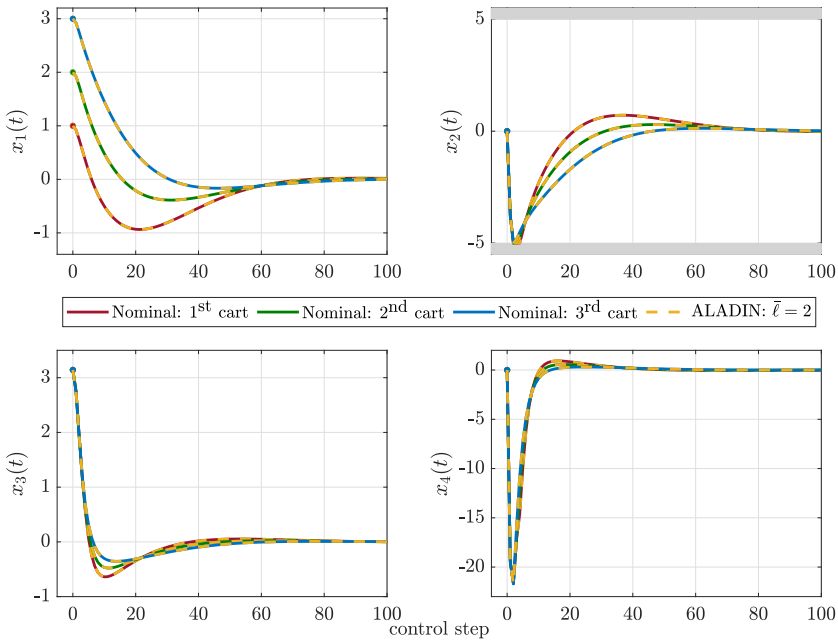


Figure 5.5: Comparison of the closed-loop state trajectories using Algorithm 5 with $\bar{\ell} = 2$ and nominal MPC. The state variables of the first, second, and third carts are shown in red, green, and blue, respectively. Only the second state variable $x_2(t)$ of each cart is subject to constraints.

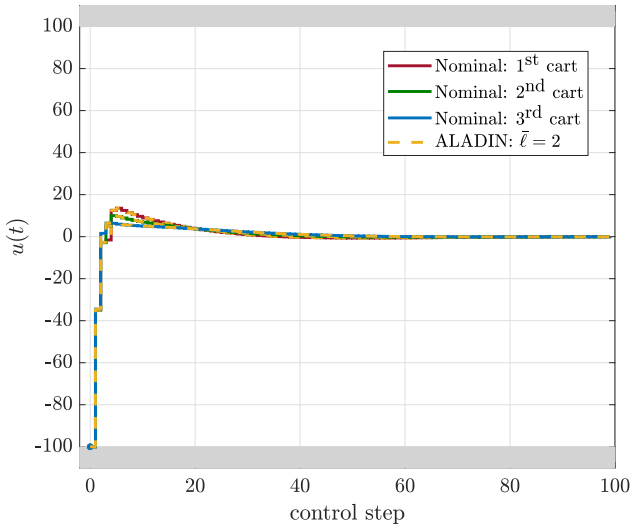


Figure 5.6: Comparison of the closed-loop control inputs obtained using Algorithm 5 with $\bar{\ell} = 2$ and the nominal MPC. The control inputs of the first, second, and third carts are depicted in red, green, and blue, respectively.

5.4 Discussion

This chapter presents a novel real-time MPC solver that integrates a relaxed, recentered logarithmic barrier method into a distributed optimization framework. The proposed approach guarantees closed-loop stability despite the suboptimality of the control actions, offering robust theoretical foundations for real-time implementation.

A key advantage of the method is its fixed-iteration scheme, which eliminates the need to evaluate stopping criteria at each sampling step. This not only simplifies implementation but also significantly reduces communication overhead compared to traditional distributed MPC strategies with variable iteration counts. Additionally, the coordination mechanism, based on the decentralized framework from Engelmann et al. (2020), can further facilitate deployment in fully decentralized settings with minimal inter-agent communication.

The effectiveness of the solver was demonstrated through simulations on a spring-damper system with coupled inverted pendulums. The results confirm that the proposed method achieves high accuracy and constraint satisfaction with only two iterations of a distributed RRLB-based MPC solver. Compared to established solvers, the proposed

algorithm achieves a runtime reduction of up to 50% while maintaining comparable control performance, confirming its suitability for real-time and resource-constrained applications.

Chapter 6

Conclusions

The main goal of this thesis was to address key challenges in the real-time implementation of MPC, with a focus on structured linear systems, large-scale systems, embedded implementation, and distributed control architectures. The presented contributions include theoretical developments, algorithm design, and practical implementations, all aimed at enabling efficient, reliable, and scalable MPC strategies under computational and communication constraints.

The first significant contribution is the introduction of a generalized stopping criterion for real-time linear-quadratic MPC with input constraints. This criterion is applicable to a broad class of linearly convergent first-order optimization methods and guarantees asymptotic closed-loop stability. Compared to algorithm-specific stopping criteria in the existing literature, the proposed criterion is method-independent and based on mild assumptions. It allows for pre-computation of the required number of iterations, thus enabling predictable and bounded computational and memory load. Numerical results using projected gradient descent and ADMM methods confirm significant reductions in iteration counts, up to 70%, while maintaining the asymptotic stability of the closed-loop system and exhibiting only marginal suboptimality.

Consequently, the thesis introduces a hybrid distributed MPC framework that combines explicit MPC, distributed optimization, and parallel computation. The novel dualization of the initial condition within Parallel MPC framework improves recursive feasibility and significantly reduces both memory usage and offline computational burden. The proposed strategy is implemented in the open-source toolbox `ParExMPC`, which demonstrates a decrease in computational demand across benchmark problems, achieving speedups of 1.2× to 3× compared to state-of-the-art solvers. Despite faster runtimes, the `ParExMPC` toolbox achieves the same level of solution accuracy as well-established solvers. Moreover, the successful implementation of the proposed framework on resource-limited hardware, such as the Arduino Uno, confirms the framework's suitability for embedded applications.

The third contribution extends the real-time distributed MPC formulation by introducing a novel dual-splitting strategy. The original MPC problem is distributed not only along the prediction horizon (temporal domain) but also across the state space (spatial domain), resulting in small-scale subproblems that can be solved efficiently. This framework is particularly beneficial for large-scale systems, where solving the original nominal MPC problem may be computationally intractable. Additionally, the proposed distributed MPC formulation incorporates a relaxed recentered logarithmic barrier (RRLB) function to handle constraints effectively. This leads to unconstrained convex QPs, which can be solved efficiently using well-established numerical methods. We have proved that the generalized stopping criterion can be applied in this setting and that the theoretical guarantees remain valid even in the presence of RRLB functions. By employing a fixed-iteration procedure, the resulting distributed RRLB-based MPC scheme offers an efficient real-time control method with fixed and predictable computational and memory requirements. This is further supported by simulation results for coupled spring-damper and inverted pendulum systems, which demonstrate a reduction in runtime of up to 50% compared to conventional approaches.

However, it is worth noting that evaluating a fixed number of iterations, as proposed in the generalized stopping criterion, requires offline tuning of the parameters in the formula, which can be demanding, particularly for large-scale systems. While more conservative parameter values can be determined with less effort, they result in a higher number of iterations, which may or may not be tractable for real-time implementation. The runtimes of the presented algorithms could also be improved by using warm-starting strategies. Moreover, both Algorithm 4 and Algorithm 5 can be implemented in a fully decentralized framework without a central coordinator, further broadening their applicability. In general, the presented concepts provide a foundation for real-time MPC that can be further extended for various purposes, such as, i.e. reference-tracking problems.

Overall, this thesis aimed to bridge the gap between theoretical guarantees and practical applications by proposing real-time MPC schemes that exploit problem structure while ensuring a fixed computational load. The resulting strategies are scalable and well-suited for deployment in large-scale systems and on embedded hardware.

Main Contributions

The main contributions of this thesis, which addresses the challenge of designing and implementing structure-exploiting linear MPC for real-time and embedded applications, are summarized below, along with the corresponding publications in which they were presented.

- **Formulation of a generalized stopping criterion for arbitrary first-order methods applied to linear MPC with input constraints.** We have formulated a generalized stopping criterion that provides fixed and predictable computational and memory requirements for solving linear MPC problems with input constraints while ensuring the asymptotic stability guarantee of the closed-loop system despite the use of suboptimal control actions. These results were published in:

  K. Fedorová – Y. Jiang – J. Oravec – C. Jones – M. Kvasnica: A Generalized Stopping Criterion for Real-Time MPC with Guaranteed Stability. In 62nd IEEE Conference on Decision and Control, IEEE, Singapore, pp. 4705–4710, 2023.

- **Enhancement of a parallelizable linear MPC strategy.** We proposed a novel dualization of the initial condition in a parallelizable MPC framework to ensure recursive feasibility. When integrated with the distributed explicit MPC framework, this modeling approach also leads to reduced memory requirements. The results were published in:

  Y. Jiang – K. Fedorová – J. Su – J. Oravec – B. Houska – C. Jones: Fast and Lightweight: A Real-Time Parallelizable MPC for Embedded Systems. European Journal of Control, vol. 83, pp. 101217, 2025.

- **Proof of global linear convergence for the ALADIN algorithm.** We provide the theoretical proof of the global linear convergence of the ALADIN algorithm applied to distributed convex nonlinear problems with separable nonlinear objectives and fixed scaling matrices. This proof has not been provided in the literature yet. The results are summarized in the publication currently under review:

  Y. Jiang – K. Fedorová – R. Schwan – J. Oravec – C. Jones: Distributed Real-Time Cooperative Model Predictive Control. IEEE Transactions on Automatic Control (under review, 3$^{\text{nd}}$ round).

- **Design of a novel MPC strategy based on spatio-temporal decomposition.** We introduced a novel approach that exploits the structure of the MPC problem and combines temporal decomposition over the prediction horizon with

spatial partitioning of the system dynamics, resulting in a distributed control scheme with reduced computational complexity and improved scalability. The results are summarized in the publication currently under review:

Y. Jiang – K. Fedorová – R. Schwan – J. Oravec – C. Jones: Distributed Real-Time Cooperative Model Predictive Control. IEEE Transactions on Automatic Control (under review, 3nd round).

- **Extension of the generalized stopping criterion to MPC problems with state constraints.** The proposed generalized stopping criterion is adapted to hold even for the MPC problems with state constraints by incorporating relaxed logarithmic barrier functions within the ALADIN framework, ensuring asymptotic closed-loop stability. The results are summarized in the publication currently under review:

  Y. Jiang – K. Fedorová – R. Schwan – J. Oravec – C. Jones: Distributed Real-Time Cooperative Model Predictive Control. IEEE Transactions on Automatic Control (under review, 3nd round).

- **Validation of the proposed strategies on benchmark systems and embedded hardware.** All methods were thoroughly tested on established benchmark problems and demonstrated in real-time on resource-constrained embedded platforms. The results were published or are currently under review:

  – For `ParExMPC` toolbox:

    Y. Jiang – K. Fedorová – J. Su – J. Oravec – B. Houska – C. Jones: Fast and Lightweight: A Real-Time Parallelizable MPC for Embedded Systems. European Journal of Control, vol. 83, pp. 101217, 2025.

  – For MPC with spatio-temporal decomposition:

    Y. Jiang – K. Fedorová – R. Schwan – J. Oravec – C. Jones: Distributed Real-Time Cooperative Model Predictive Control. IEEE Transactions on Automatic Control (under review, 3nd round).

- **Development of the open-source `ParExMPC` toolbox.** We introduced a parallelizable software tool for solving linear MPC problems, built upon the proposed framework and tailored for efficient deployment on embedded platforms. The results were published in:

  Y. Jiang – K. Fedorová – J. Su – J. Oravec – B. Houska – C. Jones: Fast and Lightweight: A Real-Time Parallelizable MPC for Embedded Systems. European Journal of Control, vol. 83, pp. 101217, 2025.

Appendix A

Publications

Papers in Journal

- Y. Jiang – **K. Fedorová** – R. Schwan – J. Oravec – C. Jones: Distributed Real-Time Cooperative Model Predictive Control. IEEE Transactions on Automatic Control (under review, 3rd round).
- Y. Jiang – **K. Fedorová** – J. Su – J. Oravec – B. Houska – C. Jones: Fast and Lightweight: A Real-Time Parallelizable MPC for Embedded Systems. European Journal of Control, vol. 83, pp. 101217, 2025.
- **K. Fedorová** – P. Bakaráč – M. Kvasnica: Agile Manoeuvres using Model Predictive Control. Acta Chimica Slovaca, no. 1, vol. 12, pp. 136–141, 2019.

Papers in Conference Proceedings

- **K. Fedorová** – Y. Jiang – J. Oravec – C. Jones – M. Kvasnica: A Generalized Stopping Criterion for Real-Time MPC with Guaranteed Stability. In 62nd IEEE Conference on Decision and Control, IEEE, Singapore, pp. 4705–4710, 2023.
- R. Kohút – E. Pavlovičová – **K. Fedorová** – J. Oravec – M. Kvasnica: Real-Time Deep-Learning-Driven Parallel MPC. In 62nd IEEE Conference on Decision and Control, IEEE, Singapore, 2023.
- T. Ábelová – R. Kohút – **K. Fedorová** – M. Kvasnica: Risk-Aware Stochastic Energy Management of Microgrid with Battery Storage and Renewables. In IFAC World Congress 2023, Yokohama, Japan, 2023.
- **K. Fedorová** – T. Ábelová – M. Kvasnica: Dynamic Power Purchase Agreement. Editor(s): R. Paulen and M. Fikar, In Proceedings of the 2023 24th Interna-

tional Conference on Process Control, IEEE, Slovak University of Technology in Bratislava, Radlinského 9, 81237, Bratislava, Slovakia, 2023.

- T. Ábelová – **K. Fedorová** – M. Kvasnica: Optimization-Based Power Distribution Method for State-of-Charge Balancing of Battery Storage Systems. Editor(s): R. Paulen and M. Fikar, In Proceedings of the 2023 24th International Conference on Process Control, IEEE, Slovak University of Technology in Bratislava, Radlinského 9, 81237, Bratislava, Slovakia, 2023.

- **K. Fedorová** – M. Kvasnica: Predictive Thermal Management of an Industrial Battery Energy Storage System. In 2022 Cybernetics & Informatics (K&I), 2022.

- **K. Fedorová** – R. Kohút – M. Kvasnica: Streamlining Active Set Method in MPC using Cache Memory. In Preprints of the 7th IFAC Conference on Nonlinear Model Predictive Control, IFAC-PapersOnLine, Bratislava, Slovakia, no. 6, vol. 54, 2021.

- R. Kohút – L. Galčíková – **K. Fedorová** – T. Ábelová – M. Bakošová – M. Kvasnica: Hidden Markov Model-based Warm-start of Active Set Method in Model Predictive Control. Editor(s): R. Paulen and M. Fikar, In Proceedings of the 23rd International Conference on Process Control, IEEE, Slovak University of Technology, 2021.

- **K. Fedorová** – P. Bakaráč – M. Kvasnica: Comparison of Two Approaches to Agile Manoeuvres via MPC. Editor(s): M. Fikar and M. Kvasnica, In Proceedings of the 22nd International Conference on Process Control, Slovak Chemical Library, Štrbské Pleso, Slovakia, 2019.

Appendix B

Curriculum vitae

Kristína Fedorová

Email: kristina.fedorova@stuba.sk
Homepage: https://www.uiam.sk/~fedorova
ORCID iD: 0000-0002-6064-4444
Year of Birth: 1996

Education

- 2020 – present PhD-study: Faculty of Chemical and Food Technology, STU in Bratislava, Study Programme: Process Control
- 2018 – 2020 Master-study: Faculty of Chemical and Food Technology, STU in Bratislava, Study Programme: Automation and Information Engineering in Chemistry and Food Industry, Graduated Summa Cum Laude
- 2015 – 2018 Bachelor-study: Faculty of Chemical and Food Technology, STU in Bratislava, Study Programme: Automation, Information Engineering and Management in Chemistry and Food Industry, Graduated Summa Cum Laude

Participation on Research Projects

European Projects

- 2022 – 2025 Fostering Opportunities Towards Slovak Excellence in Advanced Control for Smart Industries (Investigator)

APVV Research Projects

- 2021 – 2024 Energy-efficient Safe and Secure Process Control (Investigator)

VEGA Research Projects

- 2023 – 2026 Economically Efficient Predictive Control of Microgrids (Investigator)
- 2022 – 2025 Controller design methods for low-level carbon footprint process automation (Investigator)
- 2020 – 2023 Advanced Control of Energy Intensive Processes with Uncertainties in Chemical, Biochemical and Food Technologies (Investigator)
- 2019 – 2022 On-Line Tunable Explicit Model Predictive Control for Systems with a Fast Dynamics (Investigator)

Other projects

- 2023 – 2024 Enhancing Computational Efficiency in Process Control using Distributed Optimization (Internal STU scheme, Principal investigator)

International Visits and Study Stays

- 2024 EPFL – Swiss Federal Technology Institute of Lausanne, Lausanne, Switzerland
- 2022 EECI International Graduate School on Control – Control and Optimization of Autonomous Power Systems, Stockholm, Sweden
- 2021 Chinese MPC School, online

Awards

- 2020 Award of the rector of STU in Bratislava
- 2020 Award of the dean of FCHPT STU in Bratislava

- 2019 Dean's Praise for outstanding fulfillment of academic duties, extraordinary results in research and activities for the benefit of the FCHPT STU in Bratislava
- 2018 1st place in section Process control at 20th Student Scientific Conference: Chemistry and Technologies for Life
- 2018 Award of the rector of STU in Bratislava
- 2018 Award of the dean of FCHPT STU in Bratislava
- 2017 2nd place in section Process control at 19th Student Scientific Conference: Chemistry and Technologies for Life

Teaching Activities

- Optimisation of Processes and Plants (Master study, laboratory exercises)
- Laboratory Exercises of Process Control (Bachelor study, laboratory exercises)
- Process control I (Bachelor study, laboratory exercises)
- Process Control II (Bachelor study, laboratory exercises)
- Process Dynamics and Control (Master study, laboratory exercises)
- Introduction to Process Control (Bachelor study, laboratory exercises)
- Identification (Master study, laboratory exercises)
- Programming I (Bachelor study, exercises)
- Fundamentals of Matlab (Bachelor study, laboratory exercises)

Appendix C

Resumé

Táto dizertačná práca sa zaoberá návrhom algoritmických schém na riešenie optimalizačných problémov prediktívneho riadenia založenom na modeli (MPC) (Maciejowski, 2000; Mayne et al., 2000) v reálnom čase. Vyvinuté metódy sú určené na výpočet akčného zásahu pre zložité lineárne systémy, ktorých zložitosť vyplýva buď z veľkého počtu stavových premenných, alebo z dlhého predikčného horizontu. Navrhnuté prístupy sú zamerané na zníženie pamäťovej a výpočtovej náročnosti evaluácie akčného zásahu, pričom zároveň zachovávajú asymptotickú stabilitu uzavretého regulačného obvodu. Na dosiahnutie tohto cieľa sú v práci predstavené tri navzájom komplementárne prínosy, ktoré umožňujú nasadenie prediktívneho riadenia v reálnom čase aj na hardvéri s obmedzeným výpočtovým výkonom.

V prvej časti práce je predstavené všeobecné stopovacie kritérium aplikovateľné na ľubovoľnú optimalizačnú metódu prvého rádu. Metódy prvého rádu (Beck, 2017) predstavujú iteratívne algoritmy, ktoré sa v oblasti MPC často využívajú vďaka svojej nízkej výpočtovej náročnosti, jednoduchej implementácii a škálovateľnosti na problémy rôznych veľkostí. Ich nevýhodou však je, že na dosiahnutie presného riešenia typicky vyžadujú veľký počet iterácií, čo sťažuje ich nasadenie v systémoch s obmedzeným časom na výpočet.

Tieto metódy sú však robustné voči predčasnému ukončeniu, keďže iteračne konvergujú do optimálneho riešenia. Takto získané suboptimálne riešenie však nemusí zaručovať asymptotickú stabilitu uzavretého regulačného obvodu. Tento problém motivoval rozsiahly výskum v oblasti návrhu stopovacích kritérií, ktoré by aj pri suboptimálnom riešení zabezpečili stabilitu systému. V literatúre sa objavujú predovšetkým dve kategórie týchto kritérií: (i) tolerančné kritérium, ktoré zastavuje výpočet pri dosiahnutí určitej presnosti riešenia, a (ii) kritérium fixného počtu iterácií, ktoré umožňuje presnú kontrolu nad výpočtovou zložitosťou.

Tolerančné kritériá zastavujú výpočet v momente, keď je dosiahnutá preddefinovaná

úroveň presnosti riešenia ako napríklad v prácach Giselsson and Rantzer (2014) a Köhler et al. (2019). Avšak, použitie fixného počtu iterácií je výhodnejšie v reálnom čase, kde je znalosť hornej hranice výpočtového času zásadná ako uvedené v Rubagotti et al. (2014) a Liao-McPherson et al. (2022). Väčšina doterajších prác však navrhuje takéto kritériá len pre konkrétne algoritmy alebo špecifické triedy metód prvého rádu.

V tejto dizertačnej práci navrhujeme nové, všeobecne aplikovateľné stopovacie kritérium vo forme pevného počtu iterácií pre ľubovoľné metódy prvého rádu s lineárnou mierou konvergencie, predstavené v Fedorová et al. (2023). Kritérium zabezpečuje, že aj pri suboptimálnom riešení, získanom po vopred určenom počte iterácií, zostáva uzavretý regulačný systém asymptoticky stabilný. Uvedené tvrdenia sú podporené numerickými experimentmi, v ktorých preukazujeme zníženie výpočtovej náročnosti až o 70%, pri poklese kvality riadenia menšom ako 4%.

V druhej časti práce sa zameriavame na využitie predstaveného stopovacieho kritéria pri návrhu algoritmu na efektívne riešenie MPC problému. Na dosiahnutie tohto cieľa sme skombinovali dva kľúčové koncepty: (i) distribuovanú optimalizáciu a (ii) explicitné prediktívne riadenie.

Riešenie MPC problému pomocou distribuovanej optimalizácie je v súčasnosti intenzívne skúmanou a úspešne aplikovanou stratégiou. Vzhľadom na to, že štruktúra MPC problému je separovateľná z pohľadu predikčného horizontu, otvára sa priestor pre využitie rôznych paralelizovateľných nástrojov a algoritmov (Conte et al., 2016). Explicitné MPC, na druhej strane, spočíva v offline výpočte parametrických, po častiach afinných, máp, ktoré umožňujú veľmi rýchlu online evaluáciu akčného zásahu (Bemporad and Morari, 2002). Tento prístup výrazne znižuje výpočtovú náročnosť v reálnom čase. Nevýhodou explicitných metód však zostáva ich obmedzená škálovateľnosť, najmä pri veľkorozmerových systémoch alebo dlhých predikčných horizontoch, kedy rastie zložitosť offline fázy.

Táto práca preto navrhuje spojenie výhod distribuovaného a explicitného MPC. Pomocou distribúcie MPC problému získavame viacero malých, nezávislých optimalizačných úloh, ktoré zefektívňujú proces generovania explicitných máp a zároveň odstraňujú závislosť výpočtovej náročnosti na dĺžke predikčného horizontu. V nadväznosti na prácu Jiang et al. (2021), kde bol tento koncept predsatvený a simulačne overený, navrhujeme jeho rozšírenie, ktoré znižuje pamäťovú a výpočtovú náročnosť pomocou upraveného spracovania počiatočných podmienok.

Výsledkom tejto metodiky je voľne dostupný softvérový nástroj `ParExMPC`, implementovaný v jazyku `C` s používateľským rozhraním v prostredí `MATLAB`. Tento nástroj

je navrhnutý tak, aby umožňoval paralelizáciu výpočtov, čím sa zrýchľuje samotný výpočet akčného zásahu. Ako však ukazujeme, paralelizácia nie je vždy výhodná, najmä v prípade malých systémov, kde môže byť dodatočná výpočtová záťaž, spojená s paralelným spracovaním, kontraproduktívna.

V práci prezentujeme výsledky riadenia niekoľkých systémov rôznych rozmerov pomocou `ParExMPC`, pričom výpočtový čas porovnávame so známymi implicitnými MPC nástrojmi. Preukazujeme, že pri vhodne zvolenej konfigurácii, či už v paralelnom alebo sériovom režime, je možné dosiahnuť zrýchlenie výpočtu akčného zásahu minimálne o 1.2×. Okrem toho pojednávame o nasadení `ParExMPC` na hardvér s obmedzenými výpočtovými a pamäťovými zdrojmi, konkrétne na platformu Arduino Uno. Demonštrujeme, že aj na taktomto obmedzenom hardvéri je možné spoľahlivo a včas vypočítať akčný zásah, pričom zostáva dostatok výpočtovej kapacity pre ďalšie úlohy.

V poslednej časti práce sa zameriavame na teoreticky podložené zohľadnenie stavových obmedzení v kontexte navrhnutého stopovacieho kritéria, pričom analyzujeme aj štruktúru MPC problému s cieľom dosiahnuť ešte efektívnejšiu paralelizáciu výpočtu. Výsledkom tejto analýzy je nový algoritmus na riešenie MPC problému, ktorý využíva princípy relaxovanej logaritmickej bariérovej funkcie (Feller and Ebenbauer, 2018). Táto transformácia umožňuje odstránenie explicitných ohraničení zavedením penalizačného člena do účelovej funkcie, pričom výsledný optimalizačný problém ostáva konvexný, aj keď už nelineárny.

Zároveň predstavujeme novú schému priestorovo-časovej distribúcie MPC problému, kde je rozklad realizovaný nielen cez predikčný horizont (v čase), ale aj v stavovom priestore, medzi jednotlivé subsystémy. Takýto spôsob distribúcie zvyšuje škálovateľnosť riešenia a umožňuje jeho aplikáciu aj pre systémy s vysokým počtom stavových premenných. Vďaka tomu dochádza k zjednodušeniu výpočtového procesu a zrýchleniu evaluácie akčného zásahu.

Teoretická analýza potvrdzuje, že navrhnutý distribuovaný algoritmus konverguje ku globálnemu optimu. Zároveň platí, že aj pri pevne danom (obmedzenom) počte iterácií, v súlade s navrhnutým stopovacím kritériom, je zachovaná asymptotická stabilita uzavretého regulačného obvodu. Simulačné výsledky uvedené v práci ilustrujú výpočtovú náročnosť prototypovej implementácie algoritmu bez paralelizácie. Napriek tomu séria výpočtov vykonaných sekvenčne ukazuje, že navrhnutá metóda má potenciál konkurovať známym implicitným MPC riešeniam, pričom si zachováva vysokú kvalitu riadenia.

Praktická využiteľnosť tejto metódy sa rozširuje aj na systémy so sieťovou alebo

distribuovanou architektúrou, kde je vyžadovaná kooperácia medzi jednotlivými jednotkami. Ako demonštrujeme, v prípade distribúcie s centrálnou koordinačnou jednotkou je možné dosiahnuť stabilné a presné riadenie už po dvoch iteráciách. Plne decentralizovaná verzia tohto prístupu je taktiež možná, využívajúc prístupy detailne popísané v Engelmann et al. (2020).

Celá práca sa komplexne venuje riešeniu úloh prediktívneho riadenia v reálnom čase, najmä s ohľadom na jeho nasadenie na nízkovýpočtovom hardvéri, v prostredí systémov s vysokým počtom stavových premenných alebo distribuovanou architektúrou s požiadavkou na kooperáciu medzi subsystémami. Navrhnuté riešenia poskytujú teoreticky podložené garancie asymptotickej stability uzavretého regulačného obvodu, aj v prípade, keď je použitý suboptimálny akčný zásah, získaný po pevnom počte iterácií. Výsledky prezentované v tejto práci prispievajú k premosteniu medzi teoretickými garanciami a praktickou implementáciou MPC v prostrediach s obmedzeným výpočtovým výkonom. Navrhnuté prístupy tak otvárajú nové možnosti pre spoľahlivé a efektívne riadenie v reálnom čase v širokom spektre aplikácií.

Bibliography

D.A. Allan, C.N. Bates, M.J. Risbeck, and J.B. Rawlings. On the inherent robustness of optimal and suboptimal nonlinear MPC. *Systems & Control Letters*, 106:68–78, 2017.

D. Arnström, A. Bemporad, and D. Axehill. A dual active-set solver for embedded quadratic programming using recursive LDL^T updates. *IEEE Transactions on Automatic Control*, 67(8):4362–4369, 2022.

S. K. Arumugasamy and Z. Ahmad. Model predictive control (MPC) and its current issues in chemical engineering. *Chemical Engineering Communications*, 199:472–511, 04 2012.

A. Beck. *First-Order Methods in Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2017.

A. Bemporad. A quadratic programming algorithm based on nonnegative least squares with applications to embedded model predictive control. *IEEE Transactions on Automatic Control*, 61(4):1111–1116, 2015.

A. Bemporad and M. Morari. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

D.P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific Dynamic Programming and Optimal Control, Belmont, Massachusetts, 3rd edition, 2012.

J. F. Bonnans and A. Shapiro. *Perturbation analysis of optimization problems*. Springer Science & Business Media, 2013.

F. Borrelli. *Constrained Optimal Control Of Linear And Hybrid Systems*, volume 290 of *Lecture Notes in Control and Information Sciences*. Springer, 2003.

F. Borrelli, A. Bemporad, and M. Morari. Geometric algorithm for multiparametric linear programming. *Journal of Optimization Theory and Applications*, 118(3): 515–540, 2003.

F. Borrelli, A. Bemporad, and M. Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, 2011.

P. Braun, T. Faulwasser, L. Grüne, C.M. Kellett, S.R. Weller, and K. Worthmann. Hierarchical distributed ADMM for predictive control with applications in power networks. *IFAC Journal of Systems and Control*, 3:10–22, 2018.

S. Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015. ISSN 1935-8237.

C. Conte, C.N. Jones, M. Morari, and M.N. Zeilinger. Distributed synthesis and stability of cooperative distributed model predictive control for linear systems. *Automatica*, 69:117–125, 2016.

S. Di Cairano and I. V. Kolmanovsky. *Automotive Applications of Model Predictive Control*, pages 493–527. Springer International Publishing, Cham, 2019. ISBN 978-3-319-77489-3.

L. Engelmann, B. Houska, L. Grüne, and V. Hagenmeyer. Parallel distributed MPC using ALADIN with application to power system control. *IEEE Transactions on Control Systems Technology*, 28(6):2263–2270, 2020.

F. Farokhi, I. Shames, and K. H. Johansson. *Distributed MPC Via Dual Decomposition and Alternative Direction Method of Multipliers*, pages 115–131. Springer Netherlands, Dordrecht, 2014. ISBN 978-94-007-7006-5.

K. Fedorová, Y. Jiang, J. Oravec, C. N. Jones, and M. Kvasnica. A generalized stopping criterion for real-time MPC with guaranteed stability. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pages 4705–4710. IEEE, 2023.

C. Feller and C. Ebenbauer. Weight recentered barrier functions and smooth polytopic terminal set formulations for linear model predictive control. In *Amer. Control Conf.*, pages 1647–1652, 2015.

C. Feller and C. Ebenbauer. Relaxed logarithmic barrier function based model predictive control of linear systems. *IEEE Trans. Autom. Control*, 62(3):1223–1238, 2017.

C. Feller and C. Ebenbauer. Sparsity-exploiting anytime algorithms for model predictive control: A relaxed barrier approach. *IEEE Trans. Control Syst. Technol.*, 28(2): 425–435, 2018.

G. Ferrari-Trecate, L. Galbusera, D. Mignone, and R. Scattolini. Model predictive control schemes for distributed systems: A survey. In *Proceedings of the 2009 American Control Conference*, pages 4341–4346. IEEE, 2009.

H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl. qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363, 2014.

M. G. Forbes, R. S. Patwardhan, H. Hamadah, and R. B. Gopaluni. Model predictive control in industry: Challenges and opportunities. *IFAC-PapersOnLine*, 48(8): 531–538, 2015. ISSN 2405-8963. 9th IFAC Symposium on Advanced Control of Chemical Processes ADCHEM 2015.

G. Frison and M. Diehl. HPIPM: a high-performance quadratic programming framework for model predictive control. *IFAC-PapersOnLine*, 53(2):6563–6569, 2020. ISSN 2405-8963. 21st IFAC World Congress.

G. Frison, D.K.M. Kufoalor, L. Imsland, and J.B. Jørgensen. Efficient implementation of solvers for linear model predictive control on embedded devices. 2014 IEEE Conference on Control Applications (CCA), pages 1954–1959, 2014.

G. Frison, D. Kouzoupis, T. Sartor, A. Zanelli, and M. Diehl. BLASFEO: Basic linear algebra subroutines for embedded optimization. *ACM Transactions on Mathematical Software (TOMS)*, 44(4):1–30, 2018.

D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications*, 2(1):17–40, 1976. ISSN 0898-1221.

E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson. Optimal Parameter Selection for the Alternating Direction Method of Multipliers (ADMM): Quadratic Problems. *IEEE Transactions on Automatic Control*, 60(3):644–658, 2015.

P. Giselsson and A. Rantzer. Distributed model predictive control with suboptimality and stability guarantees. In *49th IEEE Conference on Decision and Control (CDC)*, pages 7272–7277, 2010.

P. Giselsson and A. Rantzer. On feasibility, stability and performance in distributed model predictive control. *IEEE Transactions on Automatic Control*, 59(4):1031–1036, 2014.

S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl. From linear to nonlinear MPC: bridging the gap via the real-time iteration. *Int. J. Control*, 93(1):62–80, 2020.

Gurobi Optimization, Inc. Gurobi Optimization, Inc. https://www.gurobi.com, 2023. Version 10.0.

M. Herceg, M. Kvasnica, C. Jones, and M. Morari. Multi-parametric toolbox 3.0. 2013 European Control Conference, pages 502–510, 2013.

M. Herceg, C. N. Jones, M. Kvasnica, and M. Morari. Enumeration-based approach to solving parametric linear complementarity problems. *Automatica*, (62):243–248, 2015.

B. Houska and Y. Jiang. *Distributed Optimization and Control with ALADIN*, page 135–163. Springer, 2021.

B. Houska, J. Frasch, and M. Diehl. An augmented Lagrangian based algorithm for distributed non-convex optimization. *SIAM J. Optim.*, 26(2):1101–1127, 2016.

B. Houska, D. Kouzoupis, Y. Jiang, and M. Diehl. Convex optimization with ALADIN. *Optimization Online preprint*, 2017.

J.L. Jerez, E.C. Kerrigan, and G.A. Constantinides. A condensed and sparse QP formulation for predictive control. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 5217–5222, 2011.

Y. Jiang, J. Oravec, B. Houska, and M. Kvasnica. Parallel MPC for linear systems with input constraints. *IEEE Transactions on Automatic Control*, 66(7):3401–3408, 2021.

Y. Jiang, K. Fedorová, R. Schwan, J. Oravec, and C. Jones. Distributed real-time cooperative model predictive control. *IEEE Transactions on Automatic Control*, (under review), 2025a.

Y. Jiang, K. Fedorová, J. Su, J. Oravec, B. Houska, and C. N. Jones. Fast and lightweight: A real-time parallelizable MPC for embedded systems. *European Journal of Control*, 83:101217, 2025b. ISSN 0947-3580.

S. Katayama, M. Murooka, and Y. Tazaki. Model predictive control of legged and humanoid robots: models and algorithms. *Advanced Robotics*, 37:1–18, 02 2023.

J. Köhler, M. A. Müller, and F. Allgöwer. Distributed model predictive control—recursive feasibility under inexact dual optimization. *Automatica*, 102:1–9, 2019. ISSN 0005-1098.

A. Kozma, J.V. Frasch, and M. Diehl. A distributed method for convex quadratic programming problems arising in optimal control of distributed systems. Proceedings of the IEEE Conference on Decision and Control (CDC), 2013.

M. Kvasnica and M. Fikar. Clipping-based complexity reduction in explicit MPC. *IEEE Transactions on Automatic Control*, 57(7):1878–1883, 2012.

M. Kvasnica, J. Löfberg, and M. Fikar. Stabilizing polynomial approximation of explicit MPC. *Automatica*, 47(4):2292–2297, 2011.

M. Kvasnica, J. Hledík, I. Rauová, and M. Fikar. Complexity reduction of explicit model predictive control via separation. *Automatica*, 49(6):1776–1781, 2013.

M. Kvasnica, B. Takács, J. Holaza, and S. Di Cairano. On region-free explicit model predictive control. 2015 54th IEEE Conference on Decision and Control (CDC), pages 3669–3674, 2015.

J. Leung, D. Liao-McPherson, and I. V. Kolmanovsky. A computable plant-optimizer region of attraction estimate for time-distributed linear model predictive control. In *2021 American Control Conference (ACC)*, pages 3384–3391, 2021.

D. Liao-McPherson, T. Skibik, J. Leung, I. Kolmanovsky, and M. M. Nicotra. An analysis of closed-loop stability for linear model predictive control based on time-distributed optimization. *IEEE Transactions on Automatic Control*, 67(5):2618–2625, 2022.

D. Limon, T. Alamo, F. Salas, and E.F. Camacho. On the stability of constrained MPC without terminal constraint. *IEEE Transactions on Automatic Control*, 51(5): 832–836, 2006.

J. M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, 2000.

MathWorks, Inc. *MATLAB Optimization Toolbox*, 2023.

D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

MOSEK ApS. MOSEK Optimization Toolbox. https://www.mosek.com, 2023. Version 10.0.

I. Nielsen and D. Axehill. Direct parallel computations of second-order search directions for model predictive control. *IEEE Transactions on Automatic Control*, 64(7):2845–2860, 2019.

R. Oberdieck, N. A. Diangelakis, I. Nascu, M. M. Papathanasiou, M. Sun, S. Avraamidou, and E. N. Pistikopoulos. On multi-parametric programming and its applications in process systems engineering. *Chemical Engineering Research and Design*, 116: 61–82, 2016. ISSN 0263-8762.

B. O'Donoghue, G. Stathopoulos, and S. Boyd. A splitting method for optimal control. *IEEE Transactions on Control Systems Technology*, 21(6):2432–2442, 2013.

OpenMP Architecture Review Board. OpenMP application program interface version 3.0, May 2008. URL http://www.openmp.org/mp-documents/spec30.pdf.

J. Oravec, Y. Jiang, B. Houska, and M. Kvasnica. Parallel explicit MPC for hardware with limited memory. In Proceedings of the 20th IFAC World Congress, Toulouse, France, pages 3356–3361, 2017.

I. Pappas, D. Kenefake, B. Burnak, S. Avraamidou, H. S. Ganesh, J. Katz, N. A. Diangelakis, and E. N. Pistikopoulos. Multiparametric programming in process systems engineering: Recent developments and path forward. *Frontiers in Chemical Engineering*, 2, 1 2021.

A. Parsi, P. Anagnostaras, A. Iannelli, and R.S. Smith. Computationally efficient robust MPC using optimized constraint tightening. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 1770–1775. IEEE, 2022.

J. B. Rawlings, D. Q. Mayne, and M. M. Diehl. Model predictive control: Theory, computation, and design. *Nob Hill Publishing*, 2017.

J.B. Rawlings, D.Q. Mayne, and M.M. Diehl. Model predictive control: Theory and design. 2009.

M. Rubagotti, P. Patrinos, and A. Bemporad. Stabilizing linear model predictive control under inexact numerical optimization. *IEEE Transactions on Automatic Control*, 59(6):1660–1666, 2014.

F. Salem and M. I. Mosaad. A comparison between MPC and optimal PID controllers: Case studies. In *Michael Faraday IET International Summit 2015*, pages 59–65, 2015.

M. Schwenzer, M. Ay, T. Bergs, and M. Abel. A real-time parallelizable MPC for embedded systems. *at - Automatisierungstechnik*, 69(5):421–432, 2021.

J. Shi, Y. Jiang, J. Oravec, and B. Houska. Parallel MPC for linear systems with state and input constraints. *IEEE Control Systems Letters*, 7:229–234, 2022.

B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, 2020.

G. Stomberg, A. Engelmann, M. Diehl, and T. Faulwasser. Decentralized real-time iterations for distributed nonlinear model predictive control. In *https://arxiv.org/pdf/2401.14898*, 2024.

J. Tarragona, A. L. Pisello, C. Fernández, A. de Gracia, and L. F. Cabeza. Systematic review on model predictive control strategies applied to active thermal energy storage systems. *Renewable and Sustainable Energy Reviews*, 149:111385, 2021. ISSN 1364-0321.

P. Tøndel and T. A. Johansen. Complexity reduction in explicit linear model predictive control. *IFAC Proceedings Volumes*, 35(1):189–194, 2002.

P. Tøndel, T.A. Johansen, and A. Bemporad. Evaluation of piecewise affine control via binary search tree. *Automatica*, 39(5):945–950, 2003.

Y. Wang, B. O'Donoghue, and S. Boyd. Approximate dynamic programming via iterated Bellman inequalities. *International Journal of Robust and Nonlinear Control*, 22(10):1134–1151, 2012.

S. J. Wright. *Primal-Dual Interior-Point Methods*. SIAM, 1997.

Y. Xi, D. Li, and S. Lin. Model predictive control — status and challenges. *Acta Automatica Sinica*, 39:222–236, 03 2013.

Z. Xianyi, W. Qian, and Z. Chothia. OpenBLAS. *URL: http://xianyi. github. io/OpenBLAS*, page 88, 2012.

Y. Yang, Y. Wang, C. Manzie, and Y. Pu. Sub-optimal MPC with dynamic constraint tightening. *IEEE Control Systems Letters*, 7:1111–1116, 2023.

M.N. Zeilinger, M. Morari, and C.N. Jones. Soft constrained model predictive control with robust stability guarantees. *IEEE Transactions on Automatic Control*, 59(5): 1190–1202, 2014.

W. Zhang and P. Shi. Distributed model predictive control using a cooperative dual decomposition. *Automatica*, 47(8):1682–1688, 2011.