Distributed and Localized Model Predictive Control

Thesis by

Carmen Amo Alonso

In Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2023
Defended May 12, 2023

ACKNOWLEDGEMENTS

I am grateful to a lot of people without whom this thesis would not have been possible. But since this is an academic piece, I will only mention here those who directly played a role in my academic development during these years.

First and foremost, I feel incredibly lucky to have had Prof. John Doyle as my advisor. When I first reached out to him, I was still an aerospace student who had as much curiosity as ignorance about control theory. In spite of this, he was incredibly supportive: he helped me switch departments, join his group, and become a CDS student. Many years later, I approached him with an even harder proposition: I wanted to apply control theory to natural language and language technologies. John was once again incredibly supportive and encouraging, and he devoted a lot of energy into helping me shape these ideas. I feel very grateful for all of this. During these years, I have enjoyed the flexibility of being his student, and learned a lot from his creative way of thinking. In no other place I would have had the freedom that led me to where I am today.

During my earlier years as a graduate student, many people helped me learn the foundations and take my first steps in research. I feel very grateful to Prof. Nikolai Matni for all his mentorship during these years. Without him, the topic of this thesis would have been very different. I am also very grateful to Prof. James Anderson, who also provided very helpful guidance and insights on my papers. I also want to thank Dr. Shih-Hao Tseng, who taught me all I know about hardware implementations. Shih-Hao is the most patient person I know; he met with me weekly for two pandemic years to help me become a better researcher and a better engineer. I am very grateful for all the time and energy that Shih-Hao devoted to my growth.

This thesis would not have been the same without the wonderful collaborations that I had during these years. First, I want to thank my colleague Lisa Li. I had the pleasure and honor to collaborate with Lisa in a big part of the papers that make this thesis. I feel very lucky to have had the chance to work alongside her, learn from her, and get her honest feedback every time. I also want to thank Fengjun Yang, who has been a great collaborator and colleague. He was always able to make our Zoom meetings very fun regardless of how tired I was. I feel also very lucky to have worked with Dimitar Ho. Our collaboration started when he once came to my office to pick up a monitor, and he ended up as a co-author of my first paper. I also want to thank all my other colleagues in John's group for giving advice and sharing moments throughout these years.

Lastly, I want to express gratitude to those who have supported me during my recent transition to applying control to the study of language, particularly Prof. Lior Pachter, Dr. Damian Blasi, and Prof. Terry Sejnowski. I also want to thank all the people who made it possible for me to be here today: Dr. Esperanza Barrera-Medrano, Prof. Ricardo Martinez-Botas, Prof. Ignacio Gomez, Prof. Isabel Perez-Grande, Prof. Michael Ortiz, Prof. Manuel Martinez-Sanchez, Prof. Carmen Guerra-Garcia, and, especially, Dr. Carlos Enol Garcia. Thank you for having given me an opportunity to gain experience in and exposure to research. Thank you for showing me how education can transform someone's life.

ABSTRACT

The power grid, the internet, a city of autonomous cars, neural networks in humans and intelligent systems, and the microbiome are just a few examples of large-scale distributed bio- and cyber-physical systems (CPS). Our reliance on these systems has been dramatically accelerating, yet we lack the principled theory to control their behavior that we have for more traditional applications such as in aerospace, chemical process, and robotic systems. Often, CPS are used in safety-critical applications and it is imperative that our control algorithms are able to robustly handle diverse constraints despite diverse uncertainties, and that they enjoy theoretical guarantees for feasibility and stability. Model Predictive Control (MPC) is the foundational method to address these challenges, and it has proven very successful in a wide variety of applications. However, most of these applications require a centralized MPC controller with poor scalability. For bio and CPS networks, its online real-time requirements quickly make communications and computing intractable. The work in this thesis responds to this need and provides a novel optimal and robust control framework based on MPC that is able to achieve stringent requirements with highly-scalable communications and computing. We show how these results extend naturally to the data-driven case where no models are available and control is based on past observations only. We also provide novel hardware implementations that exploit GPU technology to further accelerate computations. In order to achieve this, we leverage a feature of large-scale distributed systems that is often neglected: their sparsity. A major challenge of most distributed control algorithms to date is the fact that the global information exchanges that one achieves in the centralized case are hindered by the fact that these systems often exhibit great sparsity. Contrary to prior works, we take advantage of such sparsity, and illustrate that by integrating ideas from control theory, optimization and learning into this framing, we can develop a completely new set of algorithms, theoretical results and architectures to optimally control distributed cyber-physical systems for safety-critical applications. To do so, we introduce locality constraints in the formulation, which restrict each subsystem in the network to only communicate and influence a small neighborhood of subsystems as opposed to the entire network. By doing so, we achieve the following contributions:

1. *Distributed and localized synthesis and implementation of closed-loop model predictive controllers (MPC).* We present for the first time a MPC algorithm

for large-scale linear systems where both its synthesis as well as implementation can be performed in a distributed and localized way without strong assumptions. We call our algorithm Distributed and Localized Model Predictive Control (DLMPC). In this scheme, only local state and model information needs to be exchanged between subsystems for the computation and implementation of control actions. Moreover, the resulting distributed algorithms are robust to various types of additive disturbances and computations scale independent of the side of the network for the first time, making this approach scalable for arbitrary sizes of the systems.

2. *Minimally conservative guarantees for asymptotic stability and recursive feasibility.* In the existing literature, the introduction of these guarantees either led to excessive conservatism in the solution provided by the algorithms or resulted in additional computational burden while still introducing some conservatism. In this thesis we provide theoretical results and algorithms to compute theoretical guarantees for stability and feasibility of MPC. These computations can be performed offline prior to the DLMPC algorithm — so they do not add to the computational burden — and they introduce the minimal possible conservatism. By virtue of the locality constraints, these computations can be performed in a distributed and localized way for the first time, which makes computation very scalable.

3. *Globally optimal guarantees as compared with global MPC.* We provide a rigorous analysis of the optimal performance of DLMP as compared with a MPC scheme where global communication is allowed. We demonstrate that in cases where the underlying topology of the system is sparse (as is the case in most large-scale networks), the inclusion of local communication constraints does not result in a suboptimal solution. These results highlight the advantage of introducing locality constraints: while they do not impact globally optimal performance in a wide range of scenarios, they provide a more scalable and efficient solution than their global counterpart.

4. *Data-driven extension.* We extend the results in this thesis to the purely data-driven case, so no model of the system is needed. We show that all previous guarantees hold and the need for a model is fully replaced by past-trajectory data. Moreover, by virtue of locality constraints the sample complexity of the resulting algorithm is kept small and independent of the size of the whole network. This contribution makes our work applicable to real-world systems

since, given their large dimension and extreme complexity, it is often impossible to have a model of this system. The data-driven version of DLMPC makes optimal control with guarantees accessible to these systems. Preliminary comparisons are very favorable with methods based on ML and AI that lack any of the guarantees of DLMPC.

5. *Efficient hardware implementation.* We derive and demonstrate an efficient GPU implementation of DLMPC by exploiting the fact that the structure of the DLMPC problem fits well within the limitations of GPU computations. This work illustrates how locality constraints are not only beneficial in distributed settings, but also in centralized settings requiring parallelizable and efficient computations. This work presents an additional utility to locality constraints that can be exploited to create layered and efficient control architectures.

In summary, the work in this thesis provides for the first time a tractable and unifying framework for distributed control algorithms subject to constraints. DLMPC is the first online distributed control algorithm that allows for the robust, scalable, efficient and data-driven computations while enjoying theoretical guarantees and parallel hardware implementations. The framework presented in this thesis highlights the importance of considering locality constraints and opens a new frontier to develop effective layered control systems enabled by locality constraints. First, our theoretical contributions in 1, 2 and 3 solve several existing theoretical challenges in the field of distributed control. Second, our work in 4 makes these results applicable to real-world distributed systems where a model of the system is lacking. Lastly, 5 offers the possibility of leveraging our approach even in a centralized setting to massively speed up computations. Together, these different frontiers can be layered together to create effective and scalable control architectures for real-world complex and large-scale systems. This thesis shows how all of this can be achieved by leveraging the intrinsic and ubiquitous sparsity of large-scale distributed systems in technology and biology.

PUBLISHED CONTENT AND CONTRIBUTIONS

[1] J. S. Li and C. Amo Alonso, “Global performance guarantees for localized model predictive control,” Submitted to *IEEE Open Journal of Control Systems*, 2023. DOI: 10.48550/arXiv.2303.11264,
C.A.A. co-lead the conception of the project, participated in the development of the theoretical results and algorithms, helped with the conception of the experimental simulations, and revised the writing of the manuscript.

[2] C. Amo Alonso, J. S. Li, J. Anderson, and N. Matni, “Distributed and localized model predictive control. Part I: Synthesis and implementation,” *IEEE Transactions on Control of Network Systems*, pp. 1–12, 2022. DOI: 10.1109/TCNS.2022.3219770,
C.A.A. participated in the conception of the project, developed the theoretical results and algorithms, helped in the conception of the simulations, and lead the writing of the manuscript.

[3] C. Amo Alonso, J. S. Li, J. Anderson, and N. Matni, “Distributed and localized model predictive control. Part II: Theoretical guarantees,” *IEEE Transactions on Control of Network Systems*, pp. 1–11, 2022. DOI: 10.1109/TCNS.2023.3262650,
C.A.A. participated in the conception of the project, lead the development of the theoretical results and algorithms, helped in the conception of the simulations, and lead the writing of the manuscript.

[4] C. Amo Alonso and S.-H. Tseng, “Effective GPU parallelization of distributed and localized model predictive control,” pp. 199–206, 2022. DOI: 10.1109/ICCA54724.2022.9831839,
C.A.A. participated in the conception of the project, developed the theoretical results and algorithms, conceived and implemented the experimental simulations, and lead the writing of the manuscript. This paper was awarded the Best Student Paper Award.

[5] C. Amo Alonso*, F. Yang*, and N. Matni, “Data-driven distributed and localized model predictive control,” *IEEE Open Journal of Control Systems*, vol. 1, pp. 29–40, 2022. DOI: 10.1109/OJCSYS.2022.3171787,
C.A.A. participated in the conception of the project, participated in the development of the theoretical results and algorithms, conceived the experimental simulations, and participated the writing of the manuscript.

[6] C. Amo Alonso, J.S. Li, N. Matni, and J. Anderson, “Robust distributed and localized model predictive control,” *arXiv preprint arXiv:2103.14171*, 2021. DOI: 10.48550/arXiv.2103.14171,
C.A.A. participated in the conception of the project, developed the theoretical results and algorithms, conceived and implemented the experimental simulations, and lead the writing of the manuscript.

[7] J. S. Li, C. Amo Alonso, and J. C. Doyle, "Frontiers in scalable distributed control: SLS, MPC, and beyond," pp. 2720–2725, 2021. DOI: `10.23919/ACC50511.2021.9483130`,
C.A.A. participated in the conception of the project, participated in the development of the algorithms, helped in the conception of the experimental simulations, and revised the writing of the manuscript.

[8] C. Amo Alonso and N. Matni, "Distributed and localized closed-loop model predictive control via System Level Synthesis," pp. 5598–5605, 2020. DOI: `10.1109/CDC42340.2020.9303936`,
C.A.A. participated in the conception of the project, developed the theoretical results and algorithms, conceived and implemented the experimental simulations, and lead the writing of the manuscript.

[9] C. Amo Alonso, N. Matni, and J. Anderson, "Explicit distributed and localized model predictive control via System Level Synthesis," pp. 5606–5613, 2020. DOI: `10.1109/CDC42340.2020.9304349`,
C.A.A. participated in the conception of the project, developed the theoretical results and algorithms, conceived and implemented the experimental simulations, and lead the writing of the manuscript.

[10] C. Amo Alonso, D. Ho, and J.M. Maestre, "Distributed linear quadratic regulator robust to communication dropouts," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 3072–3078, 2020, 21st IFAC World Congress. DOI: `10.1016/j.ifacol.2020.12.1012`,
C.A.A. participated in the conception of the project, developed the theoretical results and algorithms, conceived and implemented the experimental simulations, and lead the writing of the manuscript.

[11] S.-H. Tseng, C. Amo Alonso, and S. Han, "System level synthesis via dynamic programming," pp. 1718–1725, 2020. DOI: `10.1109/CDC42340.2020.9304369`,
C.A.A. participated in the development of the algorithms, helped obtain the figures from simulations, and participated in the writing of the manuscript.

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

Model Predictive Control (MPC) has proven to be a successful control strategy across a wide variety of applications [1]. One of the reasons why MPC has become such a popular approach is its ability to seemingly handle hard constraints on the state and the input, which allows for setting where saturation and safety constraints need to be taken into consideration in the presence of disturbances. Moreover, MPC approaches are equipped with theoretical guarantees, such as recursive feasibility and asymptotic stability; and sufficient conditions based on terminal sets and cost functions have been established [2]. However, the need to control increasingly large-scale, distributed, and networked systems has limited its applicability to smaller-scale centralized systems.

Large-scale distributed systems, such as power grids and intelligent transport systems, are often impossible to control with a centralized controller. Even when such a centralized controller can be implemented, the high computational demand of MPC renders it impractical. Thus, efforts have been made to develop *distributed* MPC (DMPC) algorithms, wherein sub-controllers solve a local optimization problem, and potentially coordinate with other sub-controllers in the network. However, porting the ideas of centralized MPC to distributed systems is a challenging task, both theoretically and computationally. Furthermore, contemporary large-scale distributed systems such as the Internet of Things enjoy ubiquitous sensing and communication, but are locally resource constrained in terms of power consumption, memory, and computation power. If such systems are to move from passive data-collecting networks to active distributed control systems, algorithmic approaches that exploit the aforementioned advantages subject to the underlying resource constraints of the network must be developed.

Hence, the challenge of developing distributed MPC algorithms goes beyond the need for extending the centralized approaches to a distributed setting. We seek to devise a framework that exploits the aforementioned intrinsic network and communication structure and that allows for scalable algorithms independent of the size of the global system. The aim is to do so in a non-conservative and computationally efficient manner, so that DMPC can be used in real-world applications. In order to do so, additional considerations are needed to provide efficient and non-conservative theoretical guarantees together with a rigorous characterization of the impact of local communication constraints on performance. Moreover, we believe that having a data-driven approach to DMPC is paramount, since it eliminates the need for expensive system identification algorithms. Our hypothesis is that for such systems, collecting local trajectory data from a small subset of neighboring systems is a far more feasible approach than deriving the intricate and detailed system models needed by model-based control algorithms. Thus far, the field is lacking a framework for DMPC that takes into account the structure of the underlying system and provider efficient, scalable, data-driven and non-conservative solutions with theoretical guarantees.

1.1 Prior work

Most of the research on DMPC has concentrated on the collaborative scenario, where sub-controllers communicate information regarding state and control actions to synchronize their actions to achieve a common goal, usually by means of distributed optimization [3]–[10]. Since most of these approaches rely on nominal open-loop approaches, they often do not enjoy robustness guarantees [11]. To maintain robustness in the presence of additive disturbances, *closed-loop* policies are desirable. However, although many closed-loop formulations exist, they rely on strong assumptions and/or are unsuitable for distributed computation [12]–[17]. Prior to this thesis, a closed-loop DMPC algorithm that (i) computes structured feedback policies via convex optimization and (ii) can be solved at scale via distributed optimization does not currently exist in the literature.

In terms of guarantees, the majority of prior DMPC approaches rely on the use of distributed terminal costs and terminal sets to provide theoretical guarantees, similar to centralized MPC settings. However, in order to make them amenable to distributed settings, they often rely on additional structure or relaxation techniques that leads to conservative solutions [18]–[22]. A different line of work proposes to synthesize and update structured terminal sets online, which increases the computational and communication burden at each iteration while still relying on approximations that introduce conservatism [6], [23]–[27]. Prior DMPC approaches lack minimally conservative guarantees that can be computed offline in a scalable and efficient manner. Furthermore, few analyses exist as to the performance guarantees of DMPC when subject to structural constraints of the underlying system. Although local communication has been found to facilitate faster computational speed [28] and convergence [29] in offline distributed controllers, this typically comes at the cost of suboptimal global performance [30]–[33] and convergence [34]. In the realm of predictive control, communication constraints are important considerations [35], the improved computational speeds typically come at the cost of suboptimal global performance and lack of stability and convergence guarantees [36]. Prior works lack a principled study of this phenomenon that could inform the design of DMPC with structural constraints that preserve optimal performance.

In terms of data-enabled formulations, the majority of data-driven control approaches prior to this thesis that do not require a system identification step have focused on providing solutions to the linear quadratic regulator (LQR) problem [37]–[42]. Among those, the ones relying on behavioral systems theory [41] have given rise to several different data-enabled Model Predictive Control (MPC) approaches in the centralized setting [43]–[46]. Since these approaches require gathering past trajectories of the global system, their scalability and applicability in the distributed setting is hindered. Moreover, among the few recent works where data-driven approaches were applied to the distributed setting, providing theoretical guarantees for these approaches has proven difficult [47]. Prior work does not provide a solution on how to develop a *scalable* distributed MPC approach where the system model is unknown and only *local* measurements are available for each subsystem. It is important that such an approach enjoys the same theoretical guarantees of recursive feasibility and asymptotic stability as standard MPC approaches.

Lastly, applicability of MPC is often limited to slow processes given the big computational burden of solving an online optimization. This issue is further highlighted in the case of DMPC, where several optimization problems are solved per time step in iterative approaches. Standard methods to speed up computations, both algorithmic and hardware-wise, are currently not applied to DMPC appraoches. On the algorithmic side, the success of explicit MPC [48]–[50], which moves most of the burden offline and replaces an online optimization by a look-up table, fail to scale to large-dimensional systems. On the hardware side, most of the efforts are centered around achieving efficient computations by appropriately exploiting algorithmic features, and rely on hardware to simply parallelize mathematical operations. Hence, prior work on the hardware implementation of the algorithms is completely decoupled from the original system formulation, and therefore any hardware-intrinsic overhead can only be handled by using efficient programming practices. Despite the great promise of DMPC frameworks to deal with algorithmic-enhancements and hardware-intrinsic overheads in a principled manner through the problem formulation, its full potential has not been realized in the literature prior to this thesis.

1.2 Contributions and Roadmap

The goal of this thesis is to fill this gap in the literature, and provide a closed-loop DMPC framework that enjoys computationally efficient data-driven and scalable algorithms and implementations, non-conservative solutions and theoretical guarantees. In what follows, we describe the results and their organization in this thesis. For the sake of reproducibility, we also point to readily-available code to replicate all the simulations presented on this thesis.

Chapter 2: We present the Distributed Localized MPC (DLMPC) algorithm for linear time-invariant systems. We leverage the SLS framework [51] to define a novel parameterization of distributed closed-loop MPC policies such that the resulting synthesis problem is *both convex and structured*, allowing for the natural use of distributed optimization techniques. We show that by exploiting the sparsity of the underlying distributed system, we are able to distribute the computation via ADMM [52], thus allowing the online computation of closed-loop MPC policies to be carried out in a scalable localized manner. Our results apply to the nominal case as well as the robust case, and provide a unifying algorithm that applies to all cases and does not rely on simplifying approximations. Through numerical experiments, we confirm that the complexity of the subproblems solved at each subsystem scales as $O(1)$ relative to the state dimension of the full system for both the nominal and the robust case. Code to replicate the experiments can be found at `https://github.com/unstable-zeros/dl-mpc-sls/tree/master/2022_TNCS_DLMPC-Part-I`.

Chapter 3: We provide recursive feasibility and asymptotic stability guarantees for the DLMPC method. Our first key contribution is to provide the first exact, fully distributed computation of the *maximal* positive invariant set, or terminal set. We show that it can be expressed in terms of the closed-loop system responses, and when they are localized, the set is naturally structured without requiring additional assumptions. We leverage this to provide a distributed and localized offline algorithm for computation of the set. Our second key contribution is to provide the necessary additions to the original DLMPC algorithm to incorporate the coupled terminal constraint and terminal cost associated with the maximal positive invariant set–this provides the algorithm with feasibility and stability guarantees, respectively. We provide additions to the algorithm to accommodate coupled terminal constraint and cost based on a nested ADMM-based consensus algorithm. Overall, throughout all proposed algorithms, only local information exchanges within local neighborhoods are necessary, and computational complexity is independent of the global system size; each sub-controller first solves for its local portion of the terminal set, offline, then solves a local online MPC problem. We validate these results and further confirm the minimal conservatism introduced by this method through simulations. Code to replicate the experiments can be found at `https://github.com/unstable-zeros/dl-mpc-sls/tree/master/2022_TNCS_DLMPC-Part-II`.

Chapter 4: We analyze the impact of introducing locality constraints on the overall optimal performance of DMPC. First, we provide a rigorous characterization of how local communication constraints restrict (or preserve) the set of trajectories available under predictive control, and use this to provide guarantees on optimal global performance for localized MPC. Secondly, we provide an exact method for selecting an appropriate locality parameter d for localized MPC. To the best of our knowledge, these are the first results of this kind on local communication constraints. We provide simulation experiments validating our results. Code to replicate the experiments can be found at `https://github.com/flyingpeach/LocalizedMPCPerformance`.

Chapter 5: We present a data-driven version of the model-based DLMPC algorithm for linear time-invariant systems in a noise-free setting. We extend the results on data-driven SLS [46], which show that optimization problems over system-responses can be posed using only libraries of past system trajectories without explicitly identifying a system model, to the localized and distributed setting where subsystems can only collect and communicate information within a local neighborhood. In this way, the model-based DLMPC problem can equivalently be posed using only *local* libraries of past system trajectories, without explicitly identifying a system model. We then exploit this structure, together with the the separability properties of the objective function and constraints, and provide a distributed synthesis algorithm based on ADMM where only local information sharing is required. Hence, in the resulting implementation, each sub-controller solves a low-dimensional optimization problem defined over a local neighborhood, requiring only local data sharing and no system model. Since this problem is analogous to the model-based DLMPC problem, our approach directly inherits its guarantees for convergence, recursive feasibility and asymptotic stability. Through numerical experiments, we validate these results and further confirm that the complexity of the subproblems solved at each subsystem does not scale relative to the full size of the system. Code to replicate the experiments can be found at `https://github.com/unstable-zeros/dl-mpc-sls/tree/master/2022_OJCS_DataDriven-DLMPC`.

Chapter 6: We propose a GPU implementation for DLMPC that exploits the structure of the problem to overcome the computational limitations of the hardware architecture. Our first contribution is to propose an explicit solution for DLMPC. We leverage the separability of the DLMPC problem to provide explicit analytical solutions to the optimization problem solved by each subcontroller. We show that explicit solution requires just 3 partitions of the solution space per system state/input instantiation, thus making the point-location problem trivial when solving for each of the instantiations sequentially. Our second contribution is a principled parallel implementation of explicit DLMPC and overhead analysis through an appropriate distributed MPC framework that allows for local communication constraints. We exploit the potential for parallelization of this scheme in a GPU, where the GPU is not used to parallelize arithmetic computations but rather each computing thread is tasked with computing the operations corresponding to a subsystem in the network. Moreover, we demonstrate how simply applying standard parallelization techniques to the algorithm incurs unnecessary overhead, and we show that communication exchange constraints embedded in the framework allow us to explicitly deal with these hardware-intrinsic communication overheads in a principled manner. We note that the limitations in communication among the GPU computing threads resemble the communication scheme in control systems for large-networks, and we take advantage of the local communication constraints that are already included in the DLMPC algorithm. We demonstrate through simulations the effectiveness of our method. Code to replicate the experiments can be found at `https://github.com/camoalon/DLMPC-GPU`.

1.3 Notation

Lower-case and upper-case letters such as x and A denote vectors and matrices, respectively, although lower-case letters might also be used for scalars or functions (the distinction will be apparent from the context). Bracketed indices denote time-step of the real system, i.e., the system input is $u(t)$ at time t, not to be confused with x_t which denotes the predicted state x at time t. Superscripted variables with or without curly brackets, e.g., x^k or $x^{\{k\}}$, correspond to the value of x at the k^{th} iteration of a given algorithm. $\|\cdot\|_F$ denotes the Frobenius norm, and the $\dagger$ superscript denotes the pseudo-inverse of a matrix. An arrow above a matrix quantity denotes vectorization, i.e., $\vec{A}$ is the vectorization of A. Calligraphic letters such as $\mathcal{S}$ denote sets (with the exception of $\mathscr{A}$ and $\mathscr{B}$), and script letters such as $\mathfrak{c}$ denote a subset of $\mathbb{Z}^+$, e.g., $\mathfrak{c} = \{1, ..., n\}$. Square bracket notation, i.e., $[x]_i$ denotes the components of x corresponding to subsystem i, and, by extension $[x]_{j\in\mathcal{S}}$ denotes the components of x corresponding to every subsystem $j \in \mathcal{S}$. Unless otherwise stated, boldface lower and upper case letters such as $\mathbf{x}$ and $\mathbf{K}$ denote finite horizon signals and block lower triangular (causal) operators, respectively:

$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_T \end{bmatrix}, \ \mathbf{K} = \begin{bmatrix} K_0[0] & & & \\ K_1[1] & K_1[0] & & \\ \vdots & \ddots & \ddots & \\ K_T[T] & \dots & K_T[1] & K_T[0] \end{bmatrix},$$

where each x_i is an n-dimensional vector, and each $K_i[j]$ represents the value of matrix K at the j^{th} time-step computed at time i. Unless required, dimensions are not stated, and compatible dimension can be assumed. $\mathbf{K}(\mathfrak{r}, \mathfrak{c})$ denotes the submatrix of $\mathbf{K}$ composed of $\mathfrak{r}$ rows and $\mathfrak{c}$ columns, respectively. We denote the block columns of $\mathbf{K}$ by $\mathbf{K}\{1\}$,...,$\mathbf{K}\{T\}$, i.e. $\mathbf{K}\{1\} := [K_0[0]^\mathsf{T} \ \dots \ K_T[T]^\mathsf{T}]^\mathsf{T}$, and we use : to indicate the range of columns, i.e. $\mathbf{K}\{2 : T\}$ contains the block columns from the second to the last.

In some instances of Chapter 4, we alter the matrix notation for ease of readability. For instance, we denote each matrix forming $\mathbf{K}$ as $K_{i,j}$ instead of $K_i[j]$. For matrix A, $(A)_{i,:}$ denotes the i^{th} row of A, $(A)_{:,j}$ denotes the j^{th} column, and $(A)_{i,j}$ denotes the element in the i^{th} row and j^{th} column. $(A)_{i:,:}$ denotes the rows of A starting from the i^{th} row. For matrix A, $(A)_{\mathfrak{R},\mathfrak{C}}$ denotes the submatrix of A composed of the rows and columns specified by $\mathfrak{r}$ and $\mathfrak{c}$, respectively; for vector x, $(x)_{\mathfrak{r}}$ denotes the vector composed of the elements specified by $\mathfrak{r}$.

For ease of variable manipulation, we also introduce augmented variables. For any matrix Z, the corresponding augmented matrix Z^{blk} is defined as a block-diagonal matrix containing N_x copies of Z, i.e., $Z^{\text{blk}} := \text{blkdiag}(Z, \dots Z)$. For any matrix $Y = Z\Lambda$, the corresponding vectorization can be written as $\vec{Y} = Z^{\text{blk}}\,\vec{\Lambda}$.

For state x_0, the corresponding augmented state X is defined as

$$X(x_0) := \begin{bmatrix} (x_0)_1 I & (x_0)_2 I & \dots & (x_0)_{N_x} I \end{bmatrix}.$$

For notational simplicity, we write X instead of $X(x_0)$; dependence on x_0 is implicit. For any matrix Λ, $\Lambda x_0 = X\,\vec{\Lambda}$.

Chapter 2

SYNTHESIS AND IMPLEMENTATION

Abstract

In this chapter we present the Distributed and Localized Model Predictive Control (DLMPC) algorithm for large-scale linear systems. DLMPC is a distributed *closed-loop* model predictive control (MPC) scheme wherein only local state and model information needs to be exchanged between subsystems for the computation and implementation of control actions. We use the System Level Synthesis (SLS) framework to reformulate the centralized MPC problem, and show that this allows us to naturally impose localized communication constraints between sub-controllers. The structure of the resulting problem can be exploited to develop an Alternating Direction Method of Multipliers (ADMM) based algorithm that allows for distributed and localized computation of closed-loop control policies. We demonstrate that computational complexity of the subproblems solved by each subsystem in DLMPC is independent of the size of the global system. DLMPC is the first MPC algorithm that allows for the scalable computation and implementation of distributed closed-loop control policies, and deals with additive disturbances.

The content in this chapter has been published in [53].

2.1 Introduction

Model Predictive Control (MPC) has seen widespread success across many applications. However, the need to control increasingly large-scale, distributed, and networked systems has limited its applicability. Large-scale distributed systems are often impossible to control with a centralized controller, and even when such a centralized controller can be implemented, the high computational demand of MPC renders it impractical. Thus, efforts have been made to develop *distributed* MPC (DMPC) algorithms, wherein sub-controllers solve a local optimization problem, and potentially coordinate with other sub-controllers in the network.

Prior work

The majority of DMPC research has focused on the cooperative setting, where sub-controllers exchange state and control action information in order to coordinate their behavior so as to optimize a global objective, typically through distributed optimization [3]–[10]. Most of these approaches rely on nominal open-loop approaches, and while nominal MPC enjoys some intrinsic robustness [54], the resulting closed-loop can be destabilized by an arbitrary small disturbance [11]. Thus, to maintain robustness in the presence of additive disturbances, *closed-loop* policies are desirable.

Two main closed-loop MPC approaches are used. The first approach, which we use here, is to compute *dynamic* structured closed-loop policies using suitable parameterizations. This strategy was introduced by Goulart et al. [15]. More recent methods exploit Quadratic Invariance [16] and the Youla parameterization [17]. These methods allow distributed closed-loop control policies to be synthesized via convex optimization; however, the resulting optimization problem lacks structure and is not amenable to distributed optimization techniques. Thus, these methods do not scale to large systems. Similarly, recent works exploit the System Level Synthesis (SLS) parametrization to design robust MPC controllers [55]; however, it is unclear how these can be applied in the distributed setting. The alternative approach is to directly extend centralized methods (i.e., constraint tightening, tube MPC) instead of relying on a convex parametrization of distributed feedback policies. Contrary to the first approach, these methods are computationally efficient, but they require pre-computed stabilizing controllers, and often rely on strong assumptions, such as the existence of a *static structured* stabilizing controller (as in [12]) which can be NP-hard to compute [13], or decoupled subsystems (as in [14]).

Overall, though many closed-loop formulations exist, they rely on strong assumptions and/or are unsuitable for distributed computation. We seek a closed-loop

DMPC algorithm that (i) computes structured feedback policies via convex optimization and (ii) can be solved at scale via distributed optimization. To the best of our knowledge, no such algorithm exists.

Contributions

In this chapter we address this gap and present the Distributed Localized MPC (DLMPC) algorithm for linear time-invariant systems, which allows for the distributed computation of structured feedback policies with recursive feasibility and asymptotic stability guarantees. We leverage the SLS framework [51] to define a novel parameterization of distributed closed-loop MPC policies such that the resulting synthesis problem is *both convex and structured*, allowing for the natural use of distributed optimization techniques. Thanks to the nature of the SLS parametrization, this approach deals with disturbances in a straightforward manner with no additional assumptions. We show that by exploiting the sparsity of the underlying distributed system and resulting closed-loop system, as well as the separability properties [51] of commonly used objective functions and constraints, we are able to distribute the computation via ADMM [52], thus allowing the online computation of closed-loop MPC policies to be carried out in a scalable localized manner. Our results apply to the nominal case (as presented in [56]) as well as the robust case, and provide a unifying algorithm that applies to all cases and does not rely on simplifying approximations. In the resulting implementation, each sub-controller solves a low-dimensional optimization problem requiring only local communication of state and model information. Through numerical experiments, we validate these results and confirm that the complexity of the subproblems solved at each subsystem scales as $O(1)$ relative to the state dimension of the full system for both the nominal and the robust case.

2.2 Problem Formulation

Consider a discrete-time linear time-invariant (LTI) system with dynamics:

$$x(t+1) = Ax(t) + Bu(t) + w(t), \tag{2.1}$$

where $x(t) \in \mathbb{R}^n$ is the state, $u(t) \in \mathbb{R}^p$ is the control input, and $w(t) \in \mathcal{W} \subset \mathbb{R}^n$ is an exogenous disturbance. The system is composed of N interconnected subsystems (each having one or more states), so the state, control, and disturbance inputs can be suitably partitioned as $[x]_i$, $[u]_i$, and $[w]_i$ for each subsystem i, consequently inducing a compatible block structure $[A]_{ij}$, $[B]_{ij}$ in the system matrices (A, B). We model the interconnection topology of the system as a time-invariant unweighted directed graph $\mathcal{G}_{(A,B)}(\mathcal{E},\mathcal{V})$, where each subsystem i is identified with a vertex $v_i \in \mathcal{V}$ and an edge $e_{ij} \in \mathcal{E}$ exists whenever $[A]_{ij} \neq 0$ or $[B]_{ij} \neq 0$.

Example 1. *Consider the linear time-invariant system structured as a chain topology as shown in Fig. 2.1. Each subsystem i is subject to the dynamics*

$$[x(t+1)]_i = \sum_{j\in\{i,i\pm1\}} [A]_{ij}[x(t)]_j + [B]_{ii}[u(t)]_i + [w(t)]_i.$$

We choose B to be diagonal, so coupling between subsystems is defined by the A matrix–thus, the adjacency matrix of the corresponding graph $\mathcal{G}$ coincides with the support of A.

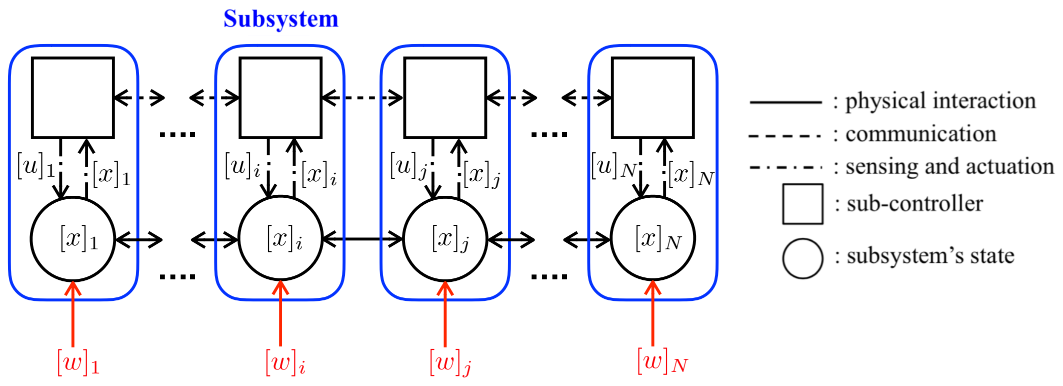

Figure 2.1: Schematic representation of a system with a chain topology.

We study the case where the control input is a model predictive controller optimizing a nominal objective and is subject to constraints on the state and the input. As is standard, at each time step τ the controller solves an optimal control problem over a finite prediction horizon of length T using the current state as the initial condition:

$$\begin{aligned} \min_{x_t,u_t,\gamma_t} \quad & \sum_{t=0}^{T-1} f_t(x_t,u_t) + f_T(x_T) \\ & x_0 = x(\tau),\ x_{t+1} = Ax_t + Bu_t + w_t, \\ \text{s.t.} \quad & x_T \in \mathcal{X}_T,\ x_t \in \mathcal{X}_t,\ u_t \in \mathcal{U}_t\ \forall w_t \in \mathcal{W}_t, \\ & t = 0,\ldots,T-1, \end{aligned} \tag{2.2}$$

where $f_t(\cdot,\cdot)$ and $f_T(\cdot)$ are closed, proper, and convex. In the nominal (i.e., noiseless) case, $\mathcal{X}_t$ and $\mathcal{U}_t$ are closed convex sets containing the origin. When noise is present, we consider polytopic sets: $\mathcal{X}_t := \{x : H_{x,t}x \le h_{x,t}\}$ and $\mathcal{U}_t := \{u : H_{u,t}u \le h_{u,t}\}$

where $H_{x,t}$, $H_{u,t}$ and $h_{x,t}$, $h_{u,t}$ are matrices and vectors of compatible size, and $\mathcal{W}_t$ is a norm-bounded or polytopic set.

In order to have a closed-loop approach to MPC, where it is possible to deal with robustness as well as have guarantees for feasibility and stability, we modify the formulation (2.2) to optimize over closed-loop policies as opposed to simply control inputs. At each time-step τ the formulation is:

$$\begin{aligned} \min_{x_t,u_t,\gamma_t} \quad & \sum_{t=0}^{T-1} f_t(x_t,u_t) + f_T(x_T) \qquad (2.3)\\ & x_0 = x(\tau),\ x_{t+1} = Ax_t + Bu_t + w_t,\\ \text{s.t.} \quad & x_T \in \mathcal{X}_T,\ x_t \in \mathcal{X}_t,\ u_t \in \mathcal{U}_t\ \forall w_t \in \mathcal{W}_t,\\ & u_t = \gamma_t(x_{0:t}, u_{0:t-1}),\ t = 0,...,T-1, \end{aligned}$$

where $\gamma_t(\cdot)$ is a measurable function of its arguments. Notice that the introduction of $\gamma_t(\cdot)$ makes the formulation be closed-loop at the expense of increasing the complexity of the formulation. In order to handle this complexity, two main approaches are used: (i) extending method from the centralized approach to the distributed setting (tube MPC, etc.) [12], [14]; or (ii) relying on a parametrization of a *dynamic* structured closed-loop policy [15]–[17]. The first approach is often hindered by strong assumptions. The method presented in this chapter belongs to the second group, but contrary to prior work, we are able to provide a distributed solution for the synthesis problem.

Our goal is to define an algorithm that allows us to solve the MPC problem (2.3) in a tractable and distributed manner while respecting local communication constraints. To achieve this goal, we impose that information exchange–as defined by the graph $\mathcal{G}_{(A,B)}(\mathcal{E},\mathcal{V})$–is localized to a subset of neighboring sub-controllers. In particular, we use the notion of a d-local information exchange constraint [51], which restricts sub-controllers to exchange their state and control actions with neighbors at most d-hops away, as measured by the communication topology $\mathcal{G}_{(A,B)}$. This notion is captured by the d-outgoing and d-incoming sets of subsystem.

Definition 1. *For a graph $\mathcal{G}(\mathcal{V},\mathcal{E})$, the d-outgoing set of subsystem i is $\boldsymbol{out}_i(d) := \{v_j \mid \boldsymbol{dist}(v_i \rightarrow v_j) \leq d \in \mathbb{N}\}$. The d-incoming set of subsystem i is $\boldsymbol{in}_i(d) := \{v_j \mid \boldsymbol{dist}(v_j \rightarrow v_i) \leq d \in \mathbb{N}\}$. Note that $v_i \in \boldsymbol{out}_i(d) \cap \boldsymbol{in}_i(d)$ for all $d \geq 0$, and $\boldsymbol{dist}(v_i \rightarrow v_j)$ denotes the distance between v_i and v_j, i.e., the number of edges in the shortest path connecting subsystems i and j.*

Although somewhat cumbersome to define, these notions are intuitive, and naturally identified with the graph topology $\mathcal{G}_{(A,B)}$ associated with the system dynamics (2.1), as we illustrate in the following example.

Example 2. *Consider a system* (2.1) *composed of* $N = 6$ *scalar subsystems, with* $B = I_6$ *and* A *matrix with support represented in Fig. 2.2(a). This induces the interconnection topology graph* $\mathcal{G}_{(A,B)}$ *illustrated in Fig. 2.2(b). The* d*-incoming and* d*-outgoing sets can be directly read off from the interaction topology. For example, for* $d = 1$*, the* 1*-hop incoming neighbors for subsystem* 5 *are subsystems* 3 *and* 4*, hence* $\boldsymbol{in}_5(1) = \{3, 4, 5\}$*; similarly,* $\boldsymbol{out}_5(1) = \{4, 5, 6\}$.

(a)

$$\mathrm{supp}(A) = \begin{bmatrix} \star & 0 & 0 & 0 & 0 & 0 \\ \star & \star & 0 & 0 & 0 & 0 \\ \star & 0 & \star & 0 & 0 & 0 \\ 0 & \star & 0 & \star & \star & 0 \\ 0 & 0 & \star & \star & \star & 0 \\ 0 & 0 & 0 & 0 & \star & \star \end{bmatrix}$$

(b)

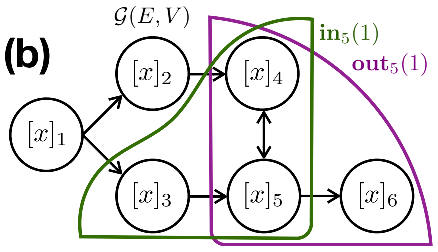

Figure 2.2: Topology of the system. (a) Support of matrix A. (b) Example of 1-incoming and 1-outgoing sets for subsystem 5.

Hence, we can enforce a d-local information exchange constraint on the MPC problem (2.3)–where the size of the local neighborhood d is a *design parameter*–by imposing that each sub-controllers policy respects

$$[u_t]_i = \gamma_t^i([x_{0:t}]_{j\in\mathbf{in}_i(d)}, [u_{0:t-1}]_{j\in\mathbf{in}_i(d)}, [A]_{j,k\in\mathbf{in}_i(d)}, [B]_{j,k\in\mathbf{in}_i(d)}) \tag{2.4}$$

for all $t = 0, \ldots, T$ and $i = 1, \ldots, N$, where γ_t^i is a measurable function of its arguments. This means that the closed-loop control policy at sub-controller i can be computed using only states, control actions, and system models collected from d-hop incoming neighbors of subsystem i in the communication topology $\mathcal{G}_{(A,B)}$. Given such an interconnection topology, suitable structural compatibility assumptions between the cost function and state, input, and information exchange constraints are necessary for both the synthesis and implementation of a localized control action at each subsystem.

Assumption 1. *In formulation* (2.3)*, objective function* f_t *is such that* $f_t(x, u) = \sum_{i=1}^{N} f_t^i([x]_j, [u]_j)$ *with* $j \in \boldsymbol{in}_i(d)$ *for local functions* f_t^i*, and state constraint sets* $\mathcal{X}_t$

are such that $x \in \mathcal{X}_t$ if and only if $[x]_{j \in \boldsymbol{in}_i(d)} \in \mathcal{X}_t^i$ $\forall i$ and $t \in \{0, ..., T\}$ for local sets $\mathcal{X}_t^i$, and idem for ($\mathcal{U}_t$, $\mathcal{U}_t^i$).

Assumption 1 imposes that whenever two subsystems are coupled through either the constraints or the objective function, they must be within the d-local regions–d-incoming and d-outgoing sets–of one another. This is a natural assumption for large structured networks where couplings between subsystems tend to occur at a local scale. We will show that under these conditions, DLMPC allows for localized synthesis and implementation of a control action at each subsystem by imposing appropriate d-local structural constraints on the *closed-loop system responses* of the system. For the remainder of this chapter we focus on developing a distributed and localized algorithmic solution and defer the design of a terminal cost and set that provides theoretical guarantees to the next chapter.

2.3 Localized MPC via System Level Synthesis

We introduce the SLS framework [51] and justify its utility in MPC problems. We show how SLS naturally allows for locality constraints [51] to be imposed on the system responses and corresponding controller implementation, and discuss how state and input constraints can be imposed in the presence of disturbances by extending the results from [57], leading to the formulation of the Distributed and Localized MPC problem in the SLS framework.

Time domain System Level Synthesis

The following is adapted from §2 of [51]. Consider the dynamics of system (2.1) and let u_t be a causal linear time-varying state-feedback controller, i.e., $u_t = K_t(x_0, x_1, ..., x_t)$ where K_t is some linear map to be designed.[1] Let Z be the block-downshift matrix,[2] and define $\hat{A} := \text{blkdiag}(A, ..., A)$ and $\hat{B} := \text{blkdiag}(B, ..., B, 0)$. Using the signal (bold) notation, we can compactly write the closed-loop behavior of system (2.1) under the feedback law $\mathbf{u} = \mathbf{K}\mathbf{x}$, over the horizon $t = 0, ..., T$, which can be entirely characterized by the system responses $\mathbf{\Phi}_x$ and $\mathbf{\Phi}_u$:

$$\begin{aligned} \mathbf{x} &= (I - Z(\hat{A} + \hat{B}\mathbf{K}))^{-1}\mathbf{w} =: \mathbf{\Phi}_x \mathbf{w} \\ \mathbf{u} &= \mathbf{K}(I - Z(\hat{A} + \hat{B}\mathbf{K}))^{-1}\mathbf{w} =: \mathbf{\Phi}_u \mathbf{w}. \end{aligned} \tag{2.5}$$

[1]Our assumption of a linear policy is without loss of generality, as an affine control policy $u_t = K_t(x_{0:t}) + v_t$ can always be written as a linear policy acting on the homogenized state $\tilde{x} = [x; 1]$.

[2]A matrix with identity matrices along its first block sub-diagonal and zeros elsewhere.

Here $\mathbf{x}$, $\mathbf{u}$ and $\mathbf{w}$ are the finite horizon signals corresponding to state, control input, and disturbance, respectively. By convention, we define the disturbance to contain the initial condition, i.e., $\mathbf{w} = [x_0^\top \; w_0^\top \; \dots \; w_{T-1}^\top]^\top$.

The approach taken by SLS is to directly parameterize and optimize over the set of achievable system responses $\{\mathbf{\Phi}_x, \mathbf{\Phi}_u\}$ from the exogenous disturbance $\mathbf{w}$ to the state $\mathbf{x}$ and the control input $\mathbf{u}$, respectively.

Theorem 1. (Theorem 2.1 of [51]) *For the system* (2.1) *evolving under the state-feedback policy* $\mathbf{u} = \mathbf{K}\mathbf{x}$*, where* $\mathbf{K}$ *is block-lower-triangular, the following are true:*

1. *The affine subspace*
$$Z_{AB}\mathbf{\Phi} := \begin{bmatrix} I - Z\hat{A} & -Z\hat{B} \end{bmatrix} \begin{bmatrix} \mathbf{\Phi}_x \\ \mathbf{\Phi}_u \end{bmatrix} = I \tag{2.6}$$
with lower-triangular $\{\mathbf{\Phi}_x, \mathbf{\Phi}_u\}$ *parameterizes all possible system responses* (2.5).

2. *For any block lower-triangular matrices* $\{\mathbf{\Phi}_x, \mathbf{\Phi}_u\}$ *satisfying* (2.6)*, the controller* $\mathbf{K} = \mathbf{\Phi}_u\mathbf{\Phi}_x^{-1}$ *achieves the desired response* (2.5) *from* $\mathbf{w} \mapsto (\mathbf{x}, \mathbf{u})$.

The SLS framework relies on part 1 of Theorem 1 to reformulate optimal control problems as a search over system responses $\mathbf{\Phi}$ lying in subspace (2.6), rather than an optimization problem over states and inputs $\{\mathbf{x}, \mathbf{u}\}$. Using parametrization (2.5), we reformulate the MPC subroutine (2.3) in terms of the system responses as

$$\begin{aligned} \min_{\mathbf{\Phi}} \quad & f(\mathbf{\Phi}\{1\}x_0) \\ \text{s.t.} \quad & Z_{AB}\mathbf{\Phi} = I, x_0 = x(\tau), \; \mathbf{\Phi}\mathbf{w} \in \mathcal{P} \; \forall \mathbf{w} \in \mathcal{W}, \end{aligned} \tag{2.7}$$

where the polytope $\mathcal{W} := \otimes_{t=0}^{T}\mathcal{W}_t$, and the polytope $\mathcal{P}$ is defined so that $\mathbf{\Phi}\mathbf{w} \in \mathcal{P}$ if and only if $x_t \in \mathcal{X}_t$, and $u_t \in \mathcal{U}_t$, for all $t = 0, ..., T-1$. Notice that $\mathcal{W}$ is defined so that it does not restrict x_0. The objective function f is defined consistent with the objective function of problem (2.3). Note that Assumption 1 directly applies to the objective function and constraint set of the SLS reformulation (2.7). We emphasize that $\mathbf{\Phi}\{1\}x_0$ appears in the objective function as it corresponds to the nominal (noise-free) state and input responses.

The equivalence between the MPC SLS problem (2.7) and the original MPC problem (2.3) stems from the well known fact–restated in terms of the SLS parameterization–that linear time-varying controllers are as expressive as nonlinear controllers over a finite horizon, given a fixed initial condition and noise realization. In fact, for a fixed initial condition and noise realization $\mathbf{w}$, any control sequence $\boldsymbol{u}(\mathbf{w}) := [u_0^\top, \dots, u_{T-1}^\top]^\top$ can be achieved by a suitable choice of feedback matrix $\boldsymbol{K}(\mathbf{w})$ such that $\boldsymbol{u}(\mathbf{w}) = \boldsymbol{K}(\mathbf{w})\mathbf{x}$ (that such a matrix always exists follows from a dimension counting argument). As this control action can be achieved by a linear-time-varying controller $\boldsymbol{K}(\mathbf{w})$, Theorem 1 states that there exists a corresponding achievable system response pair $\{\mathbf{\Phi}_x, \mathbf{\Phi}_u\}$ such that $\boldsymbol{u}(\mathbf{w}) = \mathbf{\Phi}_u \mathbf{w}$. Thus the SLS reformulation introduces no conservatism relative to open-loop MPC. We discuss the closed-loop setting at the end of this section, and show that the disturbance based parametrization [15] is a special case of ours.

Why use SLS for Distributed MPC

In the centralized setting, where both the system matrices (A, B) and the system responses $\{\mathbf{\Phi}_x, \mathbf{\Phi}_u\}$ are dense without information constraints, the SLS parameterized problem (2.7) is slightly more computationally costly than the original MPC problem (2.2), as there are now $n(n+p)T$ decision variables, as opposed to $(n+p)T$ decision variables. However, under suitable localized structural assumptions on f_t and constraint sets $\mathcal{X}_t$ and $\mathcal{U}_t$, lifting to this higher dimensional parameterization makes the problem decomposable; it allows us to take advantage of the structure of the underlying system. This allows for not only the convex synthesis of a distributed closed-loop control policy (as is similarly done in [15], [17]), but also for the solution of this convex synthesis problem to be computed using distributed optimization. Notice that the comparison being made here is between an open-loop approach (2.2) and a closed-loop one, since problem (2.7) is a reformulation of problem (2.3). Given that problem (2.3) as stated is intractable, the reformulation into the SLS parametrization as in problem (2.7) already provides an advantage.

This latter feature is one of the main contributions of this chapter. In particular, we show that the resulting number of optimization variables in the local subproblems solved at each sub-system scales as $O(d^2T)$, where d is the size of the neighborhood for each subsystem as per Definition 1 (usually $d \ll n$) and T is the time horizon; complexity is independent of the global system size n. To the best of our knowledge, this is the first distributed closed-loop MPC algorithm with such properties.

Locality in System Level Synthesis

Here we illustrate how to enforce the information sharing constraint (2.4) in the SLS framework, and how localized system responses result in a localized controller implementation.

A key advantage of using the SLS framework is that the system responses not only parametrize the closed-loop map but also provide a controller realization. In particular, the controller achieving the system responses (2.5) can be implemented as

$$\mathbf{u} = \mathbf{\Phi}_u \hat{\mathbf{w}}, \quad \hat{\mathbf{x}} = (I - \mathbf{\Phi}_x)\hat{\mathbf{w}}, \quad \hat{\mathbf{w}} = \mathbf{x} - \hat{\mathbf{x}}, \tag{2.8}$$

where $\hat{\mathbf{x}}$ is the nominal state trajectory, and $\hat{\mathbf{w}} = Z\mathbf{w}$ is a delayed reconstruction of the disturbance. The advantage of this controller implementation, as opposed to $\mathbf{u} = \mathbf{\Phi}_u \mathbf{\Phi}_x^{-1}\mathbf{x}$, is that any structure imposed on the system response $\{\mathbf{\Phi}_u, \mathbf{\Phi}_x\}$ translates directly to structure on the controller implementation (2.8). This is particularly relevant for imposing locality constraints, and we will show how locality in system responses translates into locality of the controller implementation.

We begin by defining the notion of d-localized system responses, which follows naturally from the notion of d-local information exchange constraints (2.4). They consist of system responses with suitable sparsity patterns such that the information exchange needed between subsystems to implement the controller realization (2.8) is limited to d-hop incoming and outgoing neighbors.

Definition 2. *Let $[\mathbf{\Phi}_x]_{ij}$ be the submatrix of system response $\mathbf{\Phi}_x$ describing the map from disturbance $[w]_j$ to the state $[x]_i$ of subsystem i. The map $\mathbf{\Phi}_x$ is* d-localized *if and only if for every subsystem j, $[\mathbf{\Phi}_x]_{ij} = 0 \ \forall \ i \notin \boldsymbol{out}_j(d)$. The definition for d-localized $\mathbf{\Phi}_u$ is analogous but with disturbance to control action $[u]_i$.*

When the system responses are d-localized, so is the controller implementation (2.8). In particular, by enforcing d-localized structure on $\mathbf{\Phi}_x$, only a local subset $[\hat{\mathbf{w}}]_{j \in \mathbf{in}_i(d)}$ of $\hat{\mathbf{w}}$ are necessary for subsystem i to compute its local disturbance estimate $[\hat{\mathbf{w}}]_i$, which ultimately means that only local communication is required to reconstruct the relevant disturbances for each subsystem. Similarly, if d-localized structure is imposed on $\mathbf{\Phi}_u$, then only a local subset $[\hat{\mathbf{w}}]_{j \in \mathbf{in}_i(d)}$ of the estimated disturbances $\hat{\mathbf{w}}$ is needed for each subsystem to compute its control action $[\mathbf{u}]_i$. Hence, each subsystem only needs to collect information from its d-incoming set to implement the control law (2.8), and it only needs to share information with its d-outgoing set to allow for other subsystems to implement their respective control laws. Furthermore, such locality constraints are enforced via an affine subspace constraint in the SLS formulation (2.7).

Definition 3. *A subspace $\mathcal{L}_d$ enforces a d-*locality constraint *if* $(\mathbf{\Phi}_x, \mathbf{\Phi}_u) \in \mathcal{L}_d$ *implies that $\mathbf{\Phi}_x$ is d-localized and $\mathbf{\Phi}_u$ is $(d+1)$-localized.* [3] *A system (A, B) is then d-localizable if the intersection of $\mathcal{L}_d$ with the affine space of achievable system responses* (2.6) *is non-empty.*

Remark 1. *Although d-locality constraints are always convex subspace constraints, not all systems are d-localizable. The locality diameter d can be viewed as a design parameter, and for the remainder of the chapter, we assume that there exists a $d \ll n$ such that the system (A, B) to be controlled is d-localizable. Notice that the parameter d is tuned independently of the horizon T, and captures how "far" in the interconnection topology a disturbance striking a subsystem is allowed to spread–as described in detail in [51], localized control is a spatio-temporal generalization of deadbeat control.*

Example 3. *Consider the chain in Example 1, and suppose that we enforce a* 1*-locality constraint on the system responses: then $\mathbf{\Phi}_x$ is* 1*-localized and $\mathbf{\Phi}_u$ is* 2*-localized. Due to the chain topology, this is equivalent to enforcing a tridiagonal structure on $\mathbf{\Phi}_x$ and a pentadiagonal structure on $\mathbf{\Phi}_u$. The resulting* 1*-outgoing and* 2*-incoming sets at node i are then given by* $\textbf{\textit{out}}_i(1) = \{i-1, i, i+1\}$ *and* $\textbf{\textit{in}}_i(2) = \{i-2, i-1, i, i+1, i+2\}$*, as illustrated in Fig. 2.3.*

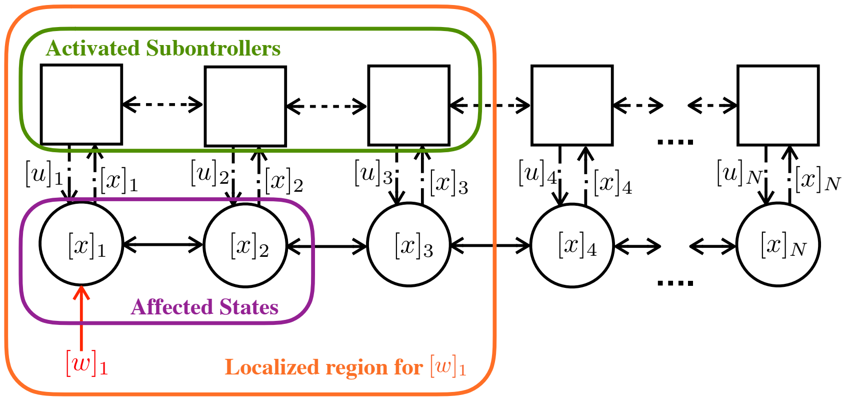

Figure 2.3: Graphical representation of the scenario in Example 3.

While it was not possible to incorporate locality (2.4) into the classical MPC formulation (2.3) in a convex and computationally tractable manner, it is straightforward

[3]Notice that we are imposing $\mathbf{\Phi}_u$ to be $(d + 1)$-localized because in order to localize the effects of a disturbance within the region of size d, the "boundary" controllers at distance $d + 1$ must take action (for more details the reader is referred to [51]).

to do so via the affine constraint ($\mathbf{\Phi_x}, \mathbf{\Phi_u} \in \mathcal{L}_d$ as per Definition 2, with the only requirement that some mild compatibility assumptions as per Assumption 1 between the cost functions, state and input constraints, and d-local information exchange constraints are satisfied. In the remainder of this section we exploit this fact when tackling robust state and input constraints, which in turn results in a problem structure that is can be solved using distributed optimization. The resulting control policies are computed using only local information.

State and input constraints in System Level Synthesis

Here we extend the method to deal with robust state and input constraints. We emphasize that the resulting SLS reformulation retains the locality structure of the original problem, which is key to enabling a distributed solution to the problem.

As previously stated, in the noisy case we restrict ourselves to polytopic constraints, such that

$$\mathcal{P} = \{[\mathbf{x}^\intercal \ \mathbf{u}^\intercal]^\intercal : \ H[\mathbf{x}^\intercal \ \mathbf{u}^\intercal]^\intercal \leq h\},$$

where $H :=$ blkdiag$(H_{x,1}, ..., H_{x,T}, H_{u,1}, ..., H_{u,T-1})$ and $h := (h_{x,1}, ..., h_{x,T}, h_{u,1}, ..., h_{u,T-1})$. We assume H and h are known. We consider two scenarios for the structure of the noise, which to distinguish from the initial condition x_0 we denote by $\boldsymbol{\delta}$, i.e., $\mathbf{w} = [x_0^\intercal \ \boldsymbol{\delta}^\intercal]^\intercal$.

Locally norm-bounded disturbance

Definition 4. *$\mathcal{W}_\sigma$ is a* separable local norm-bounded set *if for every signal $\boldsymbol{\delta} \in \mathcal{W}_\sigma$, $\|[\delta]_i\|_p \leq \sigma \ \forall i$ and $p \in \mathbb{Z}_{\geq 1}$.*

Remark 2. *Note that $\boldsymbol{\delta} \in \mathcal{W}_\sigma$ if and only if $\|\boldsymbol{\delta}\|_\infty \leq \sigma$.*

Lemma 1. *Let the noise signal belong to a separable* local *norm-bounded set $\mathcal{W}_\sigma$. Then, problem* (2.7) *with additional localization constraints has the following convex reformulation*

$$\begin{aligned} &\underset{\mathbf{\Phi}}{min} && f(\mathbf{\Phi}\{1\}x_0) && (2.9)\\ &s.t. && Z_{AB}\mathbf{\Phi} = I, \ x_0 = x(\tau), \ \mathbf{\Phi} \in \mathcal{L}_d, \ [H\mathbf{\Phi}\{1\}]_i[x_0]_i \\ & && \quad + \sum_j \sigma \left\| e_j^\intercal [H\mathbf{\Phi}\{2:T\}]_i \right\|_* e_j \leq [h]_i \end{aligned}$$

where e_j are vectors of the standard basis, and $\|\cdot\|_*$ the dual norm of $\|\cdot\|_p$.

Proof. The proof follows from a simple duality argument on the robust polytopic constraint $\mathbf{\Phi w} \in \mathcal{P}\ \forall \mathbf{w} \in \mathcal{W}$. In particular, $\mathbf{\Phi}$ has to satisfy

$$H\mathbf{\Phi}\{1\}x_0 + \max_{\{\|[\delta]_i\|_p\ \forall i\}} H\mathbf{\Phi}\{2:T\}\delta \leq h,$$

where the maximization over δ and the inequality are element-wise. Since $\mathbf{\Phi}$ is localized and the disturbance set is separable and local, this can be equivalently written as:

$$[H\mathbf{\Phi}\{1\}]_i[x_0]_i + \max_{\|[\delta]_i\|_p} [H\mathbf{\Phi}\{2:T\}]_i[\delta]_i \leq [h]_i\ \forall i.$$

And by definition of the dual norm, the equation above can be re-written as

$$[H\mathbf{\Phi}\{1\}]_i[x_0]_i + \sum_j \sigma \left\| e_j^\intercal [H\mathbf{\Phi}\{2:T\}]_i \right\|_* e_j \leq [h]_i\ \forall i.$$

□

Remark 3. *Notice that the argument of Lemma 1 can be extended to any setting for which a closed-form expression for* $\sum_j \left(\sup_{[\mathbf{w}]_i \in \mathcal{W}^i} e_j^\intercal [H\mathbf{\Phi}]_i[\mathbf{w}]_i \right) e_j \leq [h]_i$ *can be computed. Moreover, since* H *is localized block-diagonal, and* $\mathbf{\Phi}$ *is localized, this constraint can be enforced using only local information.*

Polytopic disturbance

Lemma 2. *Let the noise signal belong to a polytope, i.e.,* $\delta \in \{\delta :\ G\delta \leq g\}$, *where*

$$G := \text{blkdiag}(G_1, ..., G_T) \text{ and } g := (g_1, ..., g_T),$$

and each of the $\{G_t\}_{t=1}^T$ *are block-diagonal with structure compatible with subsystem-wise decomposition of* $\mathbf{w}$. *Then, problem* (2.7) *with* $\mathbf{\Phi} \in \mathcal{L}_d$ *is reformulated as*

$$\begin{aligned} \min_{\mathbf{\Phi}, \Xi \geq 0} \quad & f(\mathbf{\Phi}\{1\}x_0) \\ s.t. \quad & Z_{AB}\mathbf{\Phi} = I,\ x_0 = x(\tau),\ \mathbf{\Phi} \in \mathcal{L}_d,\ \Xi \in \mathcal{L}_{d_H}, \\ & H\mathbf{\Phi}\{1\}x_0 + \Xi g \leq h,\ H\mathbf{\Phi}\{2:T\} = \Xi G, \end{aligned} \tag{2.10}$$

with the constraint $\Xi \geq 0$ *satisfied component-wise, and* $\mathcal{L}_{d_H}$ *denotes the subspace of matrices with the same sparsity as* $H\mathbf{\Phi}\{2:T\}$.

Proof. The reformulation of the robust polytopic constraint follows from duality. In particular, $\mathbf{\Phi}$ has to satisfy

$$H\mathbf{\Phi}\{1\}x_0 + \max_{G\delta\leq g} H\mathbf{\Phi}\{2:T\}\boldsymbol{\delta} \leq h.$$

We study this constraint row-wise and focus on the second term on the left-hand side, which can be seen as a linear program (LP) in $\boldsymbol{\delta}$ for each row of H. Since strong duality holds, we have that each of the LPs can equivalently be replaced by its corresponding dual problem. In particular, we can solve for the k^{th} row of the second term on the left-hand side as

$$\begin{aligned} &\min_{\Xi(k,:)\geq 0} && \Xi(k,:)g \\ &\text{s.t.} && H(k,:)\mathbf{\Phi}\{2:T\} = \Xi(k,:)G, \end{aligned}$$

where the matrix operator Ξ results from stacking the dual variables from the row-wise dual LPs. Hence, the robust polytopic constraint can be replaced by

$$H\mathbf{\Phi}\{1\}x_0 + \Xi g \leq h,\ H\mathbf{\Phi}\{2:T\} = \Xi G,$$

where $\Xi \geq 0$ (satisfied component-wise) becomes a decision variable of the MPC problem. Furthermore, the constraint $H\mathbf{\Phi}\{2:T\} = \Xi G$ allows for a sparse structure on Ξ. In particular, ΞG has to have the same sparsity as $H\mathbf{\Phi}\{2:T\}$. When G is block-diagonal, it immediately follows that if $\sum_k H(i,k)\mathbf{\Phi}\{2:T\}(k,j) = 0$, then $\Xi(i,j) = 0$. Hence, Ξ lies in $\mathcal{L}_{d_H}$, and together with the dual reformulation of the robust polytopic constraint $G\boldsymbol{\delta} \leq g$, gives rise to problem (2.10). □

Remark 4. *By Assumption 1, the subspace $\mathcal{L}_{d_H}$ contains matrices with sparsity such that subsystems at most $2d$-hops away are coupled.*

These results allow us to solve the DLMPC problem (2.7) using standard convex optimization methods, and further preserves the locality structure of the original problem under the given assumptions. Imposing d-local structure on the system responses, coupled with an assumption of compatible d-local structure on the objective functions and constraints of the MPC problem (2.3), leads to a *structured* SLS MPC optimization problem (2.7). This structural compatibility in *all optimization variables, cost functions, and constraints* is the key feature that we exploit in Section 5.5 to apply distributed optimization techniques to scalably and *exactly* solve problem (2.7).[4]

[4]Notice that these locality constraints are defined in terms of the topology $\mathcal{G}_{(A,B)}$, so the structure imposed on $\mathbf{\Phi}$ will be compatible with the structure of the matrix Z_{AB} that defines constraint (2.6).

Why previous methods are not amenable to distributed solutions

While previous methods [15], [17] allow for similar structural constraints to be imposed on the controller realization through the use of either disturbance feedback or Youla parameterizations (subject to Quadratic Invariance [16] conditions), the resulting synthesis problems do not enjoy the structure needed for distributed optimization techniques to be effective. We focus on the method defined in [15], as a similar argument applies to the synthesis problem in [17]. Intuitively, the disturbance based feedback parameterization of [15] only parameterizes the closed-loop map $\mathbf{\Phi}_u$ from $\mathbf{w} \rightarrow \mathbf{u}$, and leaves the state $\mathbf{x}$ as a free variable. Hence, regardless as to what structure is imposed on the objective functions, constraints, and the map $\mathbf{\Phi}_u$, the resulting optimization problem is strongly and globally coupled because the state variable $\mathbf{x}$ is always dense. This can be made explicit by noticing that the disturbance feedback parameterization of [15] can be recovered from the SLS parameterization of Theorem 1.

Proposition 1. *The disturbance based parametrization* $(\mathbf{M}, \mathbf{v})$ *defined in Section 4 of [15] as* $\mathbf{u} = \mathbf{M}\mathbf{w} + \mathbf{v}$ *is a special case of the SLS parametrization* (2.5).

Proof. We start with the statement from Theorem 1 and in particular affine constraint (2.6). Multiplying this constraint by $\mathbf{w}$ on the right we obtain:

$$\begin{bmatrix} I - Z\hat{A} & -Z\hat{B} \end{bmatrix} \begin{bmatrix} \mathbf{\Phi}_x \\ \mathbf{\Phi}_u \end{bmatrix} \mathbf{w} = \mathbf{w},$$

where by definition of $\mathbf{\Phi}_x$, $\mathbf{x} =: \mathbf{\Phi}_x \mathbf{w}$. Hence, by Theorem 2.1 in [51] the SLS MPC subproblem

$$\begin{aligned} \min_{\mathbf{x}, \mathbf{\Phi}_u} \quad & f(\mathbf{x}, \mathbf{\Phi}_u \mathbf{w}) \\ \text{s.t.} \quad & (I - Z\hat{A})\mathbf{x} = Z\hat{B}\mathbf{\Phi}_u \mathbf{w} + \mathbf{w},\ x_0 = x(t), \\ & \mathbf{x} \in \mathcal{X}, \mathbf{\Phi}_u \mathbf{w} \in \mathcal{U}\ \forall \mathbf{w} \in \mathcal{W}, \end{aligned}$$

is equivalent to problem (2.3) when restricted to solving over linear time-varying feedback policies.

Now, if we consider nominal (disturbance free) cost, i.e., $\mathbf{w} = [x_0^\top, 0, \ldots, 0]^\top$ the problem becomes

$$\begin{aligned} \min_{\mathbf{x}, \mathbf{\Phi}_u} \quad & f(\mathbf{x}, \mathbf{\Phi}_u\{1\} x_0) \\ \text{s.t.} \quad & (I - Z\hat{A})\mathbf{x} = Z\hat{B}\mathbf{\Phi}_u \mathbf{w} + \mathbf{w},\ x_0 = x(t), \\ & \mathbf{x} \in \mathcal{X}, \mathbf{\Phi}_u \mathbf{w} \in \mathcal{U}\ \forall \mathbf{w} \in \mathcal{W}. \end{aligned}$$

Notice that by setting $\mathbf{M} = \mathbf{\Phi}_u$ and $\mathbf{v} = \mathbf{w}$, we recover the optimization problem over disturbance feedback policies $\mathbf{u} = \mathbf{M}\mathbf{w} + \mathbf{v}$ suggested in Section 4 of [15]. □

An equivalent derivation arises in the Youla-based parameterization suggested in [17]. This limits their usefulness to smaller scale examples where centralized computation of policies is feasible. In contrast, by explicitly parameterizing the additional system response $\mathbf{\Phi}_x$ from $\mathbf{w} \to \mathbf{x}$, we can naturally enforce the structure needed for distributed optimization techniques to be fruitfully applied.

2.4 Distributed AND Localized MPC Based on ADMM

We now use the previous results to reformulate the DLMPC problem (2.7) in a way that is amenable to distributed optimization techniques, and show that ADMM [52] can be used to find a solution in a distributed manner. We exploit the separability (Assumption 1), locality constraints, and the notion of row/column-wise separability (to be defined next), to solve each of the local subproblems in parallel and with d-local information only. In what follows, we restrict Assumption 1 to the case where only dynamical coupling is considered. In the next section we extend these results to all cases considered in Assumption 1, i.e., constraints and objective functions that introduce d-localized couplings. Hence, all cost function and constraints have structure:

$$f(\mathbf{x}, \mathbf{u}) = \sum_{i=1}^{N} f^i([\mathbf{x}]_i, [\mathbf{u}]_i), \text{ and}$$

$$[\mathbf{x}^\intercal \ \mathbf{u}^\intercal]^\intercal \in \mathcal{P} \text{ if and only if } \begin{bmatrix} [\mathbf{x}]_i \\ [\mathbf{u}]_i \end{bmatrix} \in \mathcal{P}^i,$$

where $\mathcal{P} = \mathcal{P}^1 \times \cdots \times \mathcal{P}^N$.

By definition of the SLS system responses (2.5), we can equivalently write these conditions in terms of $\mathbf{\Phi}$ as

$$f(\mathbf{\Phi}\{1\}x_0) = \sum_{i=1}^{N} f^i\left(\mathbf{\Phi}\{1\}(\mathfrak{r}_i, :)x_0\right), \text{ and}$$

$$\mathbf{\Phi} \in \mathcal{P} \text{ if and only if } \mathbf{\Phi}(\mathfrak{r}_i, :)x_0 \in \mathcal{P}^i \ \forall i,$$

where $\mathfrak{r}_i$ is the set of rows in $\mathbf{\Phi}$ corresponding to subsystem i, i.e., those that parametrize $[\mathbf{x}]_i$ and $[\mathbf{u}]_i$. These separability features are formalized as follows:

Definition 5. *Given the partition* $\{\mathfrak{r}_1, ..., \mathfrak{r}_k\}$*, a functional/set is* row-wise separable *if:*

- *For a functional g, $g(\mathbf{\Phi}) = \sum_{i=1}^{k} g_i\big(\mathbf{\Phi}(\mathfrak{r}_i, :)\big)$ for some functionals g_i for $i = 1, ..., k$.*
- *For a set $\mathcal{P}$, $\mathbf{\Phi} \in \mathcal{P}$ if and only if $\mathbf{\Phi}(\mathfrak{r}_i, :) \in \mathcal{P}_i$ $\forall i$ for some sets $\mathcal{P}_i$ for $i = 1, ..., k$.*

An analogous definition exists for *column-wise separable* functionals and sets [51], where the partition $\{\mathfrak{c}_1, ..., \mathfrak{c}_k\}$ entails the columns of $\mathbf{\Phi}$, i.e., $\mathbf{\Phi}(:, \mathfrak{c}_i)$.

When the objective function and all the constraints of an optimization problem are separable with respect to a partition of cardinality k, then the optimization trivially decomposes into k independent subproblems. However, this is not the case for the localized DLMPC problem (2.7), since some elements are row-wise separable while others are column-wise separable. To make it amenable to a distributed solution, we propose the following reformulation, which applies to the noise-free case as well as to both noisy cases (locally norm-bounded and polytopic) considered in the previous section:

$$\begin{aligned} \min_{\tilde{\mathbf{\Phi}}, \tilde{\mathbf{\Psi}}} \quad & f(M_1 \tilde{\mathbf{\Phi}}\{1\} x_0) \\ \text{s.t.} \quad & Z_{AB} M_2 \tilde{\mathbf{\Psi}} = I,\, x_0 = x(\tau),\ \tilde{\mathbf{\Phi}}, \tilde{\mathbf{\Psi}} \in \mathcal{L}_d, \\ & \tilde{\mathbf{\Phi}} \in \tilde{\mathcal{P}},\ \tilde{\mathbf{\Phi}} = \tilde{H} \tilde{\mathbf{\Psi}}, \end{aligned} \tag{2.11}$$

where:

- In the noise-free case:

$$\tilde{\mathbf{\Phi}} := \mathbf{\Phi}\{1\},\ \tilde{\mathbf{\Psi}} := \mathbf{\Psi}\{1\},\ M_1 = M_2 = \tilde{H} =: I,$$

$$\tilde{\mathcal{P}} = \{\mathbf{\Phi}\{1\} :\ \mathbf{\Phi}\{1\} x_0 \in \mathcal{P}\}.$$

- In the noisy case:

$$M_1 := \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}, M_2 := \begin{bmatrix} 0 & I \end{bmatrix},\ \tilde{H} := \begin{bmatrix} I & 0 \\ 0 & H \end{bmatrix},$$

  - when noise is locally bounded:

$$\tilde{\mathbf{\Phi}} := \begin{bmatrix} \mathbf{\Phi}\{1\} & 0 \\ \mathbf{\Omega}\{1\} & \mathbf{\Omega}\{2:T\} \end{bmatrix},\ \tilde{\mathbf{\Psi}} := \begin{bmatrix} \mathbf{\Psi}\{1\} & 0 \\ \mathbf{\Psi}\{1\} & \mathbf{\Psi}\{2:T\} \end{bmatrix},$$

$$\begin{aligned} \tilde{\mathcal{P}} := \{\mathbf{\Omega} :\ [\mathbf{\Omega}\{1\}]_i [x_0]_i + \sum_j \sigma \left\| e_j^{\intercal} [\mathbf{\Omega}\{2:T\}]_i e_j \right\|_* \\ \le [h]_i \forall i\}. \end{aligned}$$

– when noise is polytopic:

$$\tilde{\mathbf{\Phi}} := \begin{bmatrix} \mathbf{\Phi}\{1\} & 0 \\ \mathbf{\Omega}\{1\} & \Xi G \end{bmatrix}, \ \tilde{\mathbf{\Psi}} := \begin{bmatrix} \mathbf{\Psi}\{1\} & 0 \\ \mathbf{\Psi}\{1\} & \mathbf{\Psi}\{2:T\} \end{bmatrix},$$

$$\tilde{\mathcal{P}} := \{\mathbf{\Omega}, \Xi : \mathbf{\Omega}\{1\}x_0 + \Xi g \leq h, \ \Xi \geq 0\}.$$

The matrices $\mathbf{\Psi}$ and $\mathbf{\Omega}$ are simply duplicates of $\mathbf{\Phi}$, and Ξ is the dual variable as introduced in Lemma 2. The advantage of introducing these variables and creating the augmented variables $\tilde{\mathbf{\Phi}}$ and $\tilde{\mathbf{\Psi}}$ is that all components of problem (2.11) involving $\tilde{\mathbf{\Phi}}$ are row-wise separable, and all components involving $\tilde{\mathbf{\Psi}}$ are column-wise separable. We can easily separate these two computations via ADMM using the relationship between both variables $\tilde{H}\tilde{\mathbf{\Phi}} = \tilde{\mathbf{\Psi}}$. Furthermore, we take advantage of the structure of these subproblems and separate them with respect to a row and column partition induced by the subsystem-wise partitions of the state and control inputs, $[\mathbf{x}]_i$ and $[\mathbf{u}]_i$ for each subsystem i. Each of these row and column subproblems resulting from the distribution across subsystems can be solved independently and in parallel, where each subsystem solves for its corresponding row and column partition. Moreover, since locality constraints are imposed, the decision variables $\tilde{\mathbf{\Phi}}, \tilde{\mathbf{\Psi}}$ have a sparse structure. This means that the length of the rows and columns that a subsystem solves for is much smaller than the length of the rows and columns of $\mathbf{\Phi}$. For instance, when considering the column-wise subproblem evaluated at subsystem i, the jth row of the ith column partitions of $\mathbf{\Phi}_x$ and $\mathbf{\Phi}_u$ is nonzero only if $j \in \cup_{k\in\mathbf{out}_i(d)}\mathfrak{r}_k$ and $j \in \cup_{k\in\mathbf{out}_i(d+1)}\mathfrak{r}_k$, respectively.[5] Thus, the subproblems that subsystem i solves for are:

$$[\tilde{\mathbf{\Phi}}]_{i_r}^{k+1} = \left\{ \begin{array}{ll} \underset{[\tilde{\mathbf{\Phi}}]_{i_r}}{\text{argmin}} & f([M_1\mathbf{\Phi}]_{i_r}[x_0]_{i_r}) + \frac{\rho}{2} g_{i_r}(\tilde{\mathbf{\Phi}}, \tilde{\mathbf{\Psi}}^k, \mathbf{\Lambda}^k) \\ \text{s.t.} & [\tilde{\mathbf{\Phi}}]_{i_r} \in \tilde{\mathcal{P}}^i \cap \mathcal{L}_d, \ [x_0]_{i_r} = [x(\tau)]_{i_r} \end{array} \right\} \tag{2.12a}$$

$$[\tilde{\mathbf{\Psi}}]_{i_c}^{k+1} = \left\{ \begin{array}{ll} \underset{[\tilde{\mathbf{\Psi}}]_{i_c}}{\text{argmin}} & g_{i_c}(\tilde{\mathbf{\Phi}}^{k+1}, \tilde{\mathbf{\Psi}}, \mathbf{\Lambda}^k) \\ \text{s.t.} & [Z_{AB}]_{i_c}[M_2\tilde{\mathbf{\Psi}}]_{i_c} = [I]_{i_c} \end{array} \right\} \tag{2.12b}$$

$$[\mathbf{\Lambda}]_{i_c}^k = [\tilde{\mathbf{\Phi}}]_{i_c}^{k+1} - [\tilde{H}]_{i_c}[\tilde{\mathbf{\Psi}}]_{i_c}^{k+1} + [\mathbf{\Lambda}]_{i_c}, \tag{2.12c}$$

where the scalar ρ is the ADMM multiplier, operator $\mathbf{\Lambda}$ is the dual variable associated with ADMM, and

$$g_\bullet(\tilde{\mathbf{\Phi}}, \tilde{\mathbf{\Psi}}, \mathbf{\Lambda}) = \left\| [\tilde{\mathbf{\Phi}}]_\bullet - [\tilde{H}]_\bullet [\tilde{\mathbf{\Psi}}]_\bullet + [\mathbf{\Lambda}]_\bullet \right\|_F^2,$$

[5]An algorithm to find the relevant components for each subsystem rows and columns can be found in Appendix A of [51].

where $\bullet$ indicates i_r or i_c depending on the context. To simplify notation we denote as $[\mathbf{\Phi}]_{i_r}$ the submatrix of $\mathbf{\Phi}$ formed by the nonzero components of the relevant rows for subsystem i, $\mathfrak{r}_i$, and as $[\mathbf{\Phi}]_{i_c}$ the submatrix of $\mathbf{\Phi}$ formed by the nonzero components of the relevant columns for subsystem i, $\mathfrak{c}_i$. We use a similar bracket notation for the vectors and matrices that multiply the decision variables to indicate that those are composed from the relevant components of their global versions.

In subroutine (2.12), the computation of iterate (2.12b) can be sped up by virtue of the following Lemma:

Lemma 3. *Let $z^\star(M, v, P, q) := \underset{z}{\operatorname{argmin}} \|Mz - v\|_F^2$ s.t. $Pz = q$. Then*

$$\begin{bmatrix} z^\star \\ \mu^\star \end{bmatrix} = \begin{bmatrix} MM^\intercal & P^\intercal \\ P & 0 \end{bmatrix}^\dagger \begin{bmatrix} M^\intercal v \\ q \end{bmatrix},$$

where $\dagger$ denotes pseudo-inverse, is the optimal solution and $\mu^\star$ is the corresponding optimal Lagrange multiplier.

Proof. The proof follows from applying the KKT conditions to the optimization problem. By the stationarity condition, $M^\intercal M z^\star - M^\intercal v + P^\intercal \mu^\star = 0$, where $z^\star$ is the solution to the optimization problem and $\mu^\star$ the optimal Lagrange multiplier vector. From the primal feasibility condition, $Pz^\star = q$. Hence, $z^\star$ and $\mu^\star$ are computed as the solution to this system of two equations. □

This results in the following closed-form solution for iterate (2.12b):

$$[\mathbf{\Psi}]_{i_c}^{k+1} = z^*\Big([\tilde{H}]_{i_c}, [\tilde{\mathbf{\Phi}}]_{i_c}^{k+1} + [\tilde{\mathbf{\Lambda}}]_{i_c}^{k}, [Z_{AB}M_2]_{i_c}, [I]_{i_c}\Big). \tag{2.13}$$

Notice that the number of nonzero components for both the rows and columns is much smaller than the size of the network N since it is determined by the size of the local neighborhood d through the locality constraints. In turn, this implies that the subsystem only requires small submatrices of the system and constraint matrices $(A, B, H$, etc.) to perform the computations. Therefore, the DLMPC subroutine (2.12) can be distributed across the subsystems and solved in parallel, where each subsystem solves for a local patch of system responses. In Alg. 1 we present the DLMPC algorithm that each sub-controller executes, and where only information exchanges within d-hop neighbors take place. Alg. 1 is guaranteed to converge as shown in the next section.

Algorithm 1 Subsystem i DLMPC implementation

```
   input: $\epsilon_p, \epsilon_d, \rho > 0$.
1: Measure local state $[x(\tau)]_i$, $k \leftarrow 0$.
2: Share the measurement with neighbors in $\mathbf{out}_i(d)$. Receive the corresponding
   $[x(\tau)]_j$ from $\mathbf{in}_i(d)$ and build $[x_0]_{i_r}$.
3: Solve optimization problem (2.12a).
4: Share $[\tilde{\mathbf{\Phi}}]^{k+1}_{i_r}$ with $\mathbf{out}_i(d)$. Receive the corresponding $[\tilde{\mathbf{\Phi}}]^{k+1}_{j_r}$ from $\mathbf{in}_i(d)$ and
   build $[\tilde{\mathbf{\Phi}}]^{k+1}_{i_c}$.
5: Solve problem (2.12b) via the closed form (2.13).
6: Share $[\tilde{\mathbf{\Psi}}]^{k+1}_{i_c}$ with $\mathbf{out}_i(d)$. Receive the corresponding $[\tilde{\mathbf{\Psi}}]^{k+1}_{j_c}$ from $\mathbf{in}_i(d)$ and
   build $[\tilde{\mathbf{\Psi}}]^{k+1}_{i_r}$.
7: Perform the multiplier update step (2.12c).
8: if $\left\|[\tilde{\mathbf{\Phi}}]^{k+1}_{i_r} - [\tilde{H}]_{i_r}[\tilde{\mathbf{\Psi}}]^{k+1}_{i_r}\right\|_F \leq \epsilon_p$
   and $\left\|[\tilde{\mathbf{\Psi}}]^{k+1}_{i_r} - [\tilde{\mathbf{\Psi}}]^{k}_{i_r}\right\|_F \leq \epsilon_d$:
     Apply control action $[u_0]_i = [\Phi_{u,0}[0]]_{i_r}[x_0]_{i_r}$, and return to step 1.[6]
   else:
     Set $k \leftarrow k+1$, return to step 3.
```

Computational complexity of Alg. 1

The complexity of the algorithm is determined by update steps 3, 5 and 7. In particular, steps 5 and 7 can be directly solved in closed form, reducing their evaluation to the multiplication of matrices of dimension $O(d^2T^2)$ in the noisy case, and $O(d^2T)$ in the noise-free case. In general, step 3 requires an optimization solver where each local iterate sub-problem is over $O(d^2T^2)$ optimization variables in the noisy case, and $O(d^2T)$ in the noise-free case, subject to $O(dT)$ constraints. In certain cases, step 3 can also be computed in closed form if a proximal operator exists for the formulation [52]. This is true, for instance, if step 3 reduces to a quadratic convex cost function subject to affine equality constraints, in which case complexity reduces to $O(d^2T)$ since piece-wise closed form solutions can be computed [58]. Notice that the complexity of the subproblems is largely dependent on whether noise is considered. The reason for this is that in the noise-free case, only the first block-column of $\mathbf{\Phi}$ is considered, whereas in the presence of noise all T block-columns must be computed. Regardless, the use of locality constraints leads to a significant computational saving when $d \ll N$ (recall that both the local radius d and the time horizon T are independent design parameters). The communication complexity–as determined by steps 2, 4 and 6–is limited to the local exchange of information between d-local neighbors.

2.5 Simulation Experiments

We now apply the DLMPC algorithm to a power system inspired example. After introducing the simplified model, we present simulations[7] under different noise realizations and validate the algorithm correctness and optimal performance by comparing to a centralized algorithm, as well as its ability to achieve constraint satisfaction in the presence of noise. We further demonstrate the scalability of the proposed method by verifying different network and problem parameters (locality, network size and time horizon), and in particular show that runtime stays steady as network size increases.

System model

We begin with a two-dimensional square mesh, where we randomly determine whether each node connects to each of its neighbors with a 40% probability. The expected number of edges is $0.8 * n * (n-1)$. Each node represents a two-state subsystem that follows linearized and discretized swing dynamics

$$\begin{bmatrix} \theta(t+1) \\ \omega(t+1) \end{bmatrix}_i = \sum_{j \in \mathbf{in}_i(1)} [A]_{ij} \begin{bmatrix} \theta(t) \\ \omega(t) \end{bmatrix}_j + [B]_i [u]_i + [w]_i,$$

where $[\theta]_i$, $[\dot{\theta}]_i$, $[u]_i$ are the phase angle deviation, frequency deviation, and control action of the controllable load of bus i. The dynamic matrices are

$$[A]_{ii} = \begin{bmatrix} 1 & \Delta t \\ -\frac{k_i}{m_i}\Delta t & 1 - \frac{d_i}{m_i}\Delta t \end{bmatrix}, \; [A]_{ij} = \begin{bmatrix} 0 & 0 \\ \frac{k_{ij}}{m_i}\Delta t & 0 \end{bmatrix},$$

and $[B]_{ii} = \begin{bmatrix} 0 & 1 \end{bmatrix}^\intercal$ for all i. Parameters in bus i: m_i^{-1} (inertia inverse), d_i (damping) and k_{ij} (coupling) are randomly generated and uniformly distributed between $[0,\ 2]$, $[0.5,\ 1]$, and $[1,\ 1.5]$, respectively. The discretization step is $\Delta t = 0.2$, and $k_i := \sum_{j \in \mathbf{in}_i(1)} k_{ij}$.

In simulations, we optimize a quadratic cost $f_t(x_t, u_t) = x_t^\intercal x_t + u_t^\intercal u_t$ for all t, and start with a randomly-generated initial condition. We study three noise scenarios: noise-free, polytopic noise, and locally-bounded noise. Noise follows a uniform distribution, and in the locally-bounded case it is scaled appropriately to meet the local bounds within each d-local neighborhood (see code for details). The baseline parameter values are $d = 3,\ T = 5,\ N = 16$ (4×4 grid).

[7]Code needed to replicate these experiments is available at `https://github.com/unstable-zeros/dl-mpc-sls`; this code makes use of the SLS toolbox [59] at `https://github.com/sls-caltech/sls-code`, which includes ready-to-use MATLAB implementations of all algorithms presented in this thesis.

Optimal performance

We observe the trajectories of the closed-loop system when using DLMPC for the different noise realizations. We compare these results with the solution to the corresponding centralized MPC problem (using CVX [60]). For a randomly chosen initial condition and network topology, we plot the evolution of the two states of subsystem 4 under the different noise conditions. We use the following constraints for all i and all t: $[\theta]_i \in [-0.3,\ 0.3]$ in the noiseless case, $[\theta]_i \in [-10,\ 10]$ and $[\omega]_i \in [-0.5,\ 1.5]$ for locally-bounded noise, and $[\theta]_i \in [-3.2,\ 3.2]$ for polytopic noise.

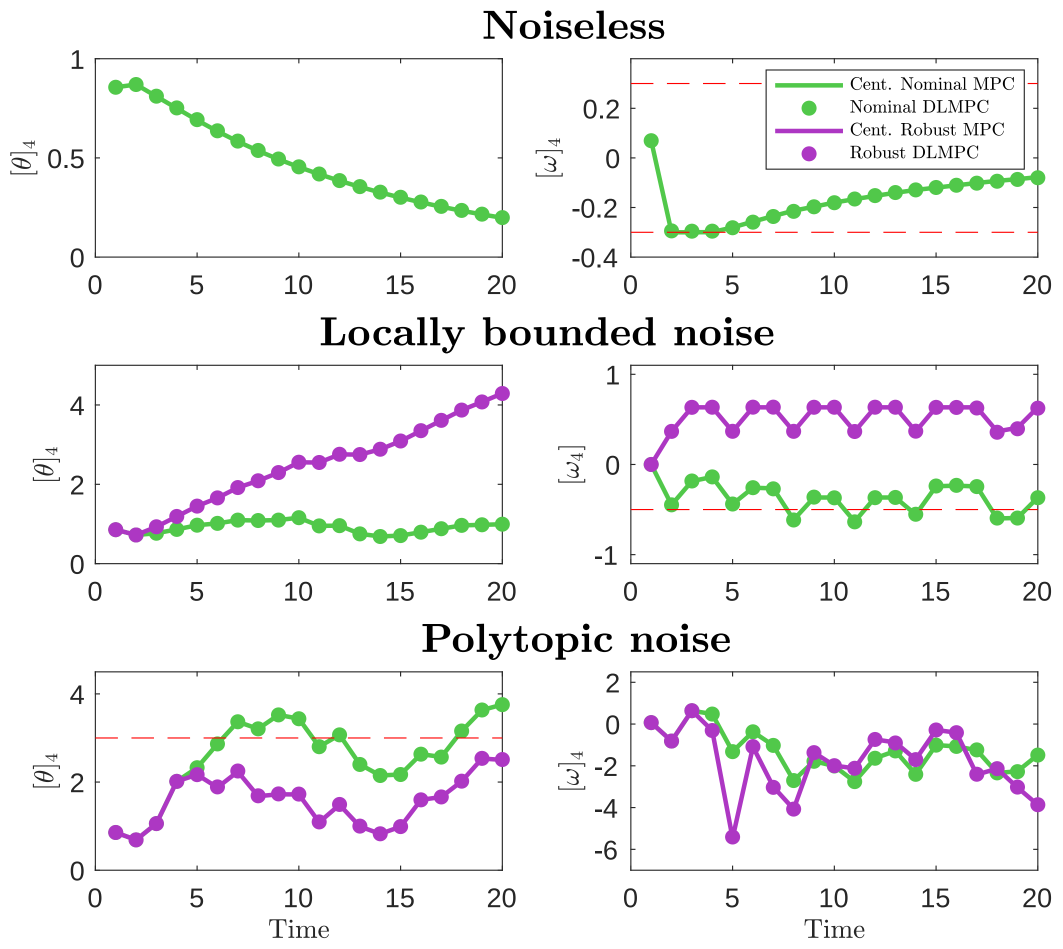

Figure 2.4: Evolution of the states of subsystem 4 under a nominal (green) and a robust (purple) controller. Red dashed line indicates upper or lower bound, respectively. Both DLMPC and centralized MPC controller yield the same result. In the absence of noise, both the nominal and robust controller lead to the same trajectory, whereas in the presence of noise (both locally-bounded and polytopic), the nominal controller leads toa violation of state bounds.

Results from simulations are summarized in Fig. 2.4. In all cases, the centralized solution coincides with the solution achieved by the DLMPC algorithm, validating the optimality of the algorithm proposed. It is worth noting that in the absence of noise, both nominal and robust DLMPC yield the same result, illustrating that the robust formulation of DLMPC is a generalization of the nominal case, but the former is more computationally efficient. In the noisy cases, nominal DLMPC violates state bounds, illustrating the need for a robust approach. In general, robust DLMPC can introduce conservatism, as it anticipates worst-case disturbances; this is more apparent for locally-bounded noise than for polytopic noise. This is because there always exists a worst-case polytopic noise realization, but there does *not* always exist a worst-case locally-bounded noise realization. Different d-local neighborhoods overlap, and are sometimes strict subsets of other neighborhoods; though the algorithm assumes a worst-case local disturbance at each neighborhood, the worst-case disturbance at some node i may be different for each d-local neighborhood containing i. Thus, there may be no noise realization that is worst-case for *every* d-local neighborhood simultaneously. Overall, locally-bounded noise formulations inherently introduce some conservatism, as they anticipate a worst-case that is generally mathematically impossible.

Computational complexity

To assess the scalability of the algorithm, we measure runtime[8] while varying different network and problem parameters: locality d, network size N[9] and time horizon T. We run 5 different simulations for each of the parameter combinations, using different realizations of the randomly chosen parameters as to provide a consistent estimation of runtimes. In this case, constraints are $[\theta]_i \in [-4,\ 4]$ and $[\omega]_i \in [-20,\ 20]$ across all scenarios.

We study the scalability of the DLMPC algorithm for each of the different computation strategies presented: noiseless, locally-bounded noise, and polytopic noise. For the sake of comparison, we include an additional computation strategy based on explicit MPC that reduces the overhead in the noiseless case by replacing the optimization solver by a piece-wise solution (we discuss this in detail in Chapter 6). We observe the behavior for these four different strategies in Fig. 6.7, and we note that the computation strategy determines the order of magnitude of the run-

[8]Runtime is measured after the first iteration, so that all the iterations for which runtime is measured are warm-started.

[9]To increase network size, we vary the size of the grid over 4×4 (32 states), 6×6 (72 states), 8×8 (128 states), and 11×11 (242 states) grid sizes.

time, ranging from $10^{-3} - 10^{-2}s$ in the noiseless case–depending on whether or not explicit solutions are used–to $10^{-1}s$ in the locally-bounded case, and around $1 - 10s$ when polytopic noise is considered. This difference is expected, and is explained by the size of the decision variables across different scenarios, i.e., $\tilde{\mathbf{\Phi}}$ has a much larger dimension in the polytopic case than in the noiseless case. Despite this difference in order of magnitude, the trends observed are the same for each of the different scenarios, and radically different from that observed when using a centralized solver–ranging from $10^{-2}s$ to $1s$–where runtime does significantly increase with the size of the network. In contrast, when using Alg. 1 runtime barely increases with the size of the network, and the slight increase in runtime–likely due to the larger coordination needed–does not seem to be significant and appears to level off for sufficiently large networks. These observations are consistent with those of [61] where the same trend was noted. In contrast, runtime appears to increase with time horizon, and more notably with locality region size. This is also expected, as according to our complexity analysis the number of variables in a DLMPC subproblem scales as $O(d^2T^2)$ for the robust cases and as $O(d^2T)$ for the nominal case. It is well-known that a larger time horizon, while providing an improvement in performance, can be detrimental for computational overhead. The same is true for locality size, which is much smaller than the size of the network. Although a larger localized region d can lead to improved performance, as a broader set of subsystems can coordinate their actions directly, it also leads to an increase in computational complexity and communication overhead. Thus by choosing the smallest localization parameter d such that acceptable performance is achieved, the designer can tradeoff between computational complexity and closed-loop performance. This further highlights the importance of exploiting the underlying structure of the dynamics, which allow us to enforce locality constraints on the system responses, and consequently, on the controller implementation.

2.6 Conclusion

We defined and analyzed a *closed-loop* Distributed and Localized MPC algorithm. By leveraging the SLS framework, we were able to enforce information exchange constraints by imposing locality constraints on the system responses. We further showed that when locality is combined with mild assumptions on the separability structure of the objective functions and constraints of the problem, an ADMM based solution to the DLMPC subproblems can be implemented that requires only local information exchange and system models, making the approach suitable for

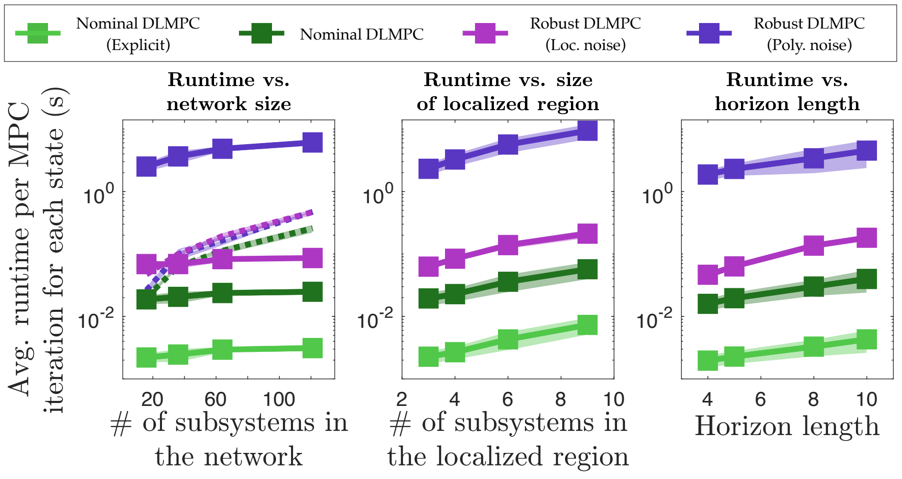

Figure 2.5: Average runtime per DLMPC iteration with network size (left), locality parameter (middle), and time horizon (right). The lines are the mean values and the shaded areas show the values within one standard deviation. Since computations are parallelized across subsystems, runtime is measured on a subsystem, normalized per state, and averaged out after the MPC controller is finished. In the case of increasing network sizes, an equivalent runtime measure was made using a centralized solver and results are presented with dashed lines, where the color used matches the corresponding scenario.

large-scale distributed systems. This is the first DMPC algorithm that allows for the distributed synthesis of closed-loop policies.

Given that the presented algorithm relies on multiple exchanges of information between the subsystems, how communication loss affects the closed-loop performance of the algorithm is an interesting question. Although a formal analysis is left as future research, the work done in [62] suggests that it would be possible to slightly modify the proposed ADMM-based scheme to make it robust to unreliable communication links. Moreover, only two types of noise were explored in this chapter. Results in polytopic containment [63] could enable a convex DLMPC reformulation of other noise scenarios. Finally, it is of interest to extend these results to information exchange topologies defined with both sparsity and delays–while the SLS framework naturally allows for delay to be imposed on the implementation structure of a distributed controller, it is less clear how to incorporate such constraints in a distributed optimization scheme.

Chapter 3

RECURSIVE FEASIBILITY, CONVERGENCE, AND ASYMPTOTIC STABILITY GUARANTEES

Abstract

In this chapter, we provide recursive feasibility, convergence, and asymptotic stability guarantees for DLMPC. We leverage the System Level Synthesis framework to express the maximal positive robust invariant set for the closed-loop system and its corresponding Lyapunov function in terms of the closed-loop system responses. We use the invariant set as the terminal set for DLMPC, and show that this guarantees feasibility with minimal conservatism. We use the Lyapunov function as the terminal cost, and show that this guarantees stability. We provide fully distributed and localized algorithms to compute the terminal set offline, and also provide necessary additions to the online DLMPC algorithm to accommodate coupled terminal constraint and cost. In all algorithms, only local information exchange is necessary, and computational complexity is independent of the global system size–we demonstrate this analytically and experimentally. This is the first distributed MPC approach that provides minimally conservative yet fully distributed guarantees for recursive feasibility and asymptotic stability in the nominal and robust settings.

The content in this chapter has been published in [64].

3.1 Introduction

Ensuring recursive feasibility and asymptoptic stability for MPC is a well-studied topic in the centralized setting [1], and sufficient conditions based on terminal sets and cost functions have been established [2]. Porting these ideas to distributed systems is a challenging task, both theoretically and computationally. High computational demand, limited and local communication, and coupling among subsystems prevents the use of techniques from the centralized setting. Thus, efforts have been made to develop theoretical guarantees for distributed MPC.

Prior work

The majority of distributed MPC approaches rely on the use of distributed terminal costs and terminal sets to provide theoretical guarantees. In order to obtain structure in the terminal cost, standard methods rely on Lyapunov stability results, often combined with relaxation techniques to make them amenable to distributed settings [18]–[20]. For terminal sets, proposed distributed methods are often limited by the coupling among subsystems–this often leads to small terminal sets that result in too conservative solutions (see for example [21], [22] and references therein). In order to overcome these issues, several approaches have recently been proposed to synthesize structured terminal sets with adaptive properties, i.e., local terminal sets defined as the sub-level set of a structured Lyapunov function, which change at each iteration in order to avoid unnecessary conservatism [6], [23]–[27]. These approaches successfully design structured robust positive invariant sets and reduce conservatism; however, they require online updates of the terminal set at each MPC iteration, which increase the controller's overall computational complexity and communication overhead. Moreover, the imposed structure unavoidably leads to a possibly small approximation of the maximal control invariant set (the least conservative option for a terminal set). To move away from structured sets and costs, a data-driven approach was recently developed in [10], where locally collected data are used to construct local control invariant sets and costs that provide guarantees. However, this approach is mainly limited to iterative control tasks, and conservatism of the terminal cost and set only reduces asymptotically as the system collects data. Online computation and refinements of the terminal set are also key to this approach.

Given the state of the art, our goal is to design a distributed MPC approach with *distributed* and *minimally conservative* feasibility and stability guarantees. We seek a distributed MPC algorithm with (i) a maximal positive invariant terminal set, and (ii) a fully distributed and scalable *offline* algorithm to compute this set. This will allow us to use the associated Lyapunov function of the terminal set as the terminal cost, and requires no explicit a priori structural assumptions. No method satisfying these requirement currently exists in the literature.

Contributions

We provide recursive feasibility and asymptotic stability guarantees for the Distributed Localized MPC (DLMPC) method presented in Chapter 2, for nominal and robust MPC (robust to additive polytopic or locally norm-bounded disturbances). This consists of two key contributions. ***Our first key contribution*** is to provide the first exact, fully distributed computation of the *maximal* positive invariant set, or terminal set. We show that it can be expressed in terms of the closed-loop system responses, as defined in the System Level Synthesis (SLS) framework [51]. Then, we show that when the closed-loop system is localized, the set is naturally structured without requiring additional assumptions; we leverage this to provide a distributed and localized offline algorithm for computation of the set. ***Our second key contribution*** is to provide the necessary additions to the original DLMPC algorithm to incorporate the coupled terminal constraint and terminal cost (i.e., global Lyapunov function [65]) associated with the maximal positive invariant set–this provides the algorithm with feasibility and stability guarantees, respectively. We provide additions to the algorithm to accommodate coupled terminal constraint and cost; this is done by using a nested Alternating Direction Method of Multipliers (ADMM)-based consensus algorithm. ***Overall***, throughout all proposed algorithms, only local information exchanges within local neighborhoods are necessary, and computational complexity is independent of the global system size; each sub-controller first solves for its local portion of the terminal set, offline, then solves a local online MPC problem. We validate these results and further confirm the minimal conservatism introduced by this method through simulations.

3.2 Problem Formulation

In Chapter 2, we showed how (3.1) can be both synthesized and implemented in a distributed and localized way. To do this, we performed a SLS reformulation of equation (2.12) and applied an ADMM distributed optimization technique, resulting in Algorithm 1. Under this framework, each subsystem requires only local information to synthesize and implement its local sub-controller and determine the local control action. This is made possible by imposing appropriate d-local structural constraints $\mathcal{L}_d$ on the *closed-loop system responses* of the system $\mathbf{\Phi}_x$ and $\mathbf{\Phi}_u$, which become the decision variables of the MPC problem. However, no explicit treatment was given to the terminal set $\mathcal{X}_T$ and terminal cost f_T in subroutine, and as a result the proposed framework lacks feasibility, and stability guarantees. Convergence guarantees of the algorithm were also not discussed.

In this chapter, we explicitly tackle the formulation and synthesis of the terminal set $\mathcal{X}_T$ and terminal cost f_T, and we leverage these to provide the aforementioned feasibility and stability guarantees. To do so, we start with the DLMPC formulation (2.12) presented in Chapter 2, but make the terminal set $\mathcal{X}_T$ and terminal cost f_T explicit in the formulation. Hence, the DLMPC subroutine over time horizon T at time τ is as follows:

$$\begin{aligned} \min_{\mathbf{\Phi}} \quad & f(\mathbf{\Phi}\{1\}x_0) + f_T(\mathbf{\Phi}\{1\}x_0) \\ \text{s.t.} \quad & Z_{AB}\mathbf{\Phi} = I,\ x_0 = x(\tau),\ \mathbf{\Phi} \in \mathcal{L}_d, \\ & \mathbf{\Phi}\mathbf{w} \in \mathcal{P},\ \mathbf{\Phi}_{x,T}\mathbf{w} \in \mathcal{X}_T,\ \forall \mathbf{w} \in \mathcal{W}, \end{aligned} \tag{3.1}$$

where $\mathbf{\Phi} := \begin{bmatrix}\mathbf{\Phi}_x^\intercal & \mathbf{\Phi}_u^\intercal\end{bmatrix}^\intercal$, $f_t(\cdot,\cdot)$ and $f_T(\cdot)$ are closed, proper, and convex cost functions, and $\mathcal{P}$ is defined so that $\mathbf{\Phi}\mathbf{w} \in \mathcal{P}$ if and only if $\mathbf{x} \in \mathcal{X}$, and $\mathbf{u} \in \mathcal{U}$. By convention, we define the disturbance to contain the initial condition, i.e., $\mathbf{w} = [x_0^\intercal\ w_0^\intercal\ \dots\ w_{T-1}^\intercal] =: [x_0^\intercal\ \boldsymbol{\delta}^\intercal]$ and $\mathcal{W}$ is defined over $\boldsymbol{\delta}$ so that it does not restrict x_0. In the nominal case, i.e., $w_t = 0\ \forall t$, $\mathcal{P}$ and $\mathcal{X}_T$ are closed and convex sets containing the origin. When noise is present, we restrict ourselves to polytopic sets only: $\mathcal{P} := \{[\mathbf{x}^\intercal\ \mathbf{u}^\intercal]^\intercal : H[\mathbf{x}^\intercal\ \mathbf{u}^\intercal]^\intercal \leq h\}$, and consider different options for set $\mathcal{W}$:

- Polytopic set: $\boldsymbol{\delta} \in \{\boldsymbol{\delta} : G\boldsymbol{\delta} \leq g\}$.
- Locally norm-bounded: $[\boldsymbol{\delta}]_i \in \{\boldsymbol{\delta} : \|\boldsymbol{\delta}\|_p \leq \sigma\}\ \forall i = 1, ..., N$ and $p \geq 1$.

In the remainder of this chapter, we address all of these gaps: we provide theoretical guarantees by selecting an appropriate terminal set and cost, and present distributed and localized algorithms that perform the necessary computations. Additionally, Algorithm 1 was developed under the assumption that subsystems are not coupled. We want to augment the algorithm to accommodate coupling as per Assumption 1.

3.3 Feasibility and Stability Guarantees

Theoretical guarantees for the DLMPC problem (3.1) are now derived. First, we describe a maximal positive invariant set using an SLS-style parametrization; recursive feasibility for DLMPC is guaranteed by using this set as the terminal set. We also use this set to construct a terminal cost to guarantee asymptotic stability for the nominal setting and input-to-state stability (ISS) for the robust setting. Convergence results from the ADMM literature are used to establish convergence guarantees.

Feasibility guarantees

Recursive feasibility guarantees for the DLMPC problem (3.1) are given by the following lemma:

Lemma 4. *Let the terminal set $\mathcal{X}_T$ for the DLMPC problem* (3.1) *be of the form*

$$\mathcal{X}_T := \{x_0 \in \mathbb{R}^n : \mathbf{\Phi} \begin{bmatrix} x_0^\intercal & \boldsymbol{\delta}^\intercal \end{bmatrix}^\intercal \in \mathcal{P} \; \forall \boldsymbol{\delta} \in \mathcal{W}\}, \tag{3.2}$$

for some $\mathbf{\Phi}$ satisfying $Z_{AB}\mathbf{\Phi} = I$. Then recursive feasibility is guaranteed for the DLMPC problem (3.1)*. Moreover, $\mathcal{X}_T$ is the maximal robust positive invariant set for the closed-loop described by $\mathbf{\Phi}$.*

Proof. First, we show that $\mathcal{X}_T$ is the maximal robust positive invariant set. By Algorithm 10.4 in [1], the set

$$\mathcal{S} := \bigcap_{k=0}^{\infty} \mathcal{S}_k, \quad \text{with} \quad \mathcal{S}_0 = \mathcal{X},$$

$$\mathcal{S}_k = \{x \in \mathbb{R}^n : \; Ax + Bu + w \in \mathcal{S}_{k-1} \; \forall w \in \mathcal{W}\}$$

is the maximal robust positive invariant set for the closed-loop system (2.1) with some fixed input $u \in \mathcal{U}$. We show by induction that $\mathcal{S}_k$ can also be written as

$$\mathcal{S}_k = \{x_0 \in \mathbb{R}^n : \; x_k \in \mathcal{X} \; \forall w_{0:k-1} \in \mathcal{W}\} \tag{3.3}$$

for some sequence $u_{0:k-1} \in \mathcal{U}$. The base case at $k = 1$ is trivially true, since $\mathcal{S}_0 = \mathcal{X}$. For the inductive step, assume that equation (3.3) holds at k. Then, at $k = 1$,

$$\begin{aligned} \mathcal{S}_{k+1} &= \left\{x \in \mathbb{R}^n : \; Ax + Bu + w \in \mathcal{S}_k \; \forall w \in \mathcal{W}\right\} \\ &= \Big\{x_0 \in \mathbb{R}^n : \; x := Ax_0 + Bu + w \; s.t. \\ &\qquad A^{k+1}x + \sum_{j=0}^{k} A^j (Bu_j + w) \in \mathcal{X} \; \forall w \in \mathcal{W}\Big\} \\ &= \left\{x_0 \in \mathbb{R}^n : \; x_{k+1} \in \mathcal{X}, \; \forall w_{0:k} \in \mathcal{W}\right\} \end{aligned}$$

for some sequence of inputs $u_{0:k} \in \mathcal{U}$. This implies that $\mathcal{S}$ can be written as

$$\mathcal{S} = \{x_0 \in \mathbb{R}^n : \; \mathbf{x} \in \mathcal{X} \; \forall w \in \mathcal{W}\} \text{ for some } \mathbf{u} \in \mathcal{U}.$$

From the definition of $\mathbf{\Phi}$, this implies directly that

$$\mathcal{S} = \{x_0 \in \mathbb{R}^n : \mathbf{\Phi} \begin{bmatrix} x_0^\intercal & \delta^\intercal \end{bmatrix} \in \mathcal{P} \; \forall \delta \in \mathcal{W}\} := \mathcal{X}_T,$$

for some $\mathbf{\Phi}$ satisfying $Z_{AB}\mathbf{\Phi} = I$. This constraint automatically enforces that the resulting closed-loop is feasible, i.e., that $\mathbf{u} = \mathbf{\Phi}_u \begin{bmatrix} x_0^\intercal & \delta^\intercal \end{bmatrix}$ exists; furthermore, all inputs $\mathbf{u}$ can be captured by this closed-loop parametrization since it parametrizes a linear time-varying controller over a finite time horizon [51]. Hence, $\mathcal{X}_T$ is the maximal robust positive invariant set for the closed-loop system as defined by $\mathbf{\Phi}$.

Next, we show that imposing $\mathcal{X}_T$ as the terminal set of the DLMPC problem (3.1) guarantees recursive feasibility. Notice that by definition, $\mathcal{X}_T$ is not only a robust positive invariant set but also a robust *control* invariant set. We can directly apply the proofs from Theorem 12.1 in [1] to the robust setting–recursive feasibility is guaranteed if the terminal set is control invariant, which $\mathcal{X}_T$ is. □

Remark 5. *Recursive feasibility is guaranteed by a* control *invariant $\mathcal{X}_T$. In the ideal case, we want $\mathcal{X}_T$ to be the maximal robust control invariant set, in order to minimize conservatism introduced by $\mathcal{X}_T$ in* (3.1). *However, this set generally lacks sparsity, violates Assumption 1, and is not amenable for inclusion in our distributed and localized algorithm. For this reason, we use the maximal robust* positive *invariant set in Lemma 4. Here, the choice of $\mathbf{\Phi}$ is critical in determining how much conservatism $\mathcal{X}_T$ will introduce. As suggested in §12 of [1], we choose $\mathbf{\Phi}$ corresponding to the unconstrained closed-loop system. In the interests of distributed synthesis and implementation, we additionally enforce $\mathbf{\Phi}$ to have localized structure. We discuss how this $\mathbf{\Phi}$ is computed in §3.4, and demonstrate that the resulting terminal set $\mathcal{X}_T$ introduces no conservatism in §3.5.*

By Assumption 1, constraint sets are localized, i.e., $\mathcal{X} = \mathcal{X}^1 \cap \ldots \cap \mathcal{X}^N$, where $x \in \mathcal{X}$ if and only if $[x]_{\mathbf{in}_i(d)} \in \mathcal{X}^i$ for all i (and idem for $\mathcal{U}$). Moreover, we can use the constraint $\mathcal{L}_d$ to enforce that the system response $\mathbf{\Phi}$ is localized. This implies that the set $\mathcal{X}_T$ is also localized:

$$\mathcal{X} = \mathcal{X}_T^1 \cap \ldots \cap \mathcal{X}_T^N, \quad \text{where}$$

$$\mathcal{X}_T^i = \{[x_0]_{\mathbf{in}_i(d)} \in \mathbb{R}^{[n]_i} : [\mathbf{\Phi}]_{\mathbf{in}_i(d)} \begin{bmatrix} x_0^\intercal & \delta^\intercal \end{bmatrix}_{\mathbf{in}_i(2d)} \in \mathcal{P}^i \quad \forall [\delta]_{\mathbf{in}_i(2d)} \in \mathcal{W}^{\mathbf{in}_i(d)}\}$$

and $x \in \mathcal{X}_T$ if and only if $[x]_{\mathbf{in}_i(d)} \in \mathcal{X}_T^i$ for all i. We will show in §3.4 that this allows for a localized and distributed computation of the terminal set $\mathcal{X}_T$.

Stability guarantees

Stability guarantees for the DLMPC problem (3.1) are given by the following lemma:

Lemma 5. *Consider system* (2.1) *subject to the MPC law* (3.1)*, where:*

1. *The cost f is continuous and positive definite.*
2. *The set $\mathcal{P}$ contains the origin and is closed.*
3. *$\mathcal{X}_T$ is defined by* (3.2)*.*
4. *$f_T(x) = inf\{\eta \geq 0 : \ x \in \eta\mathcal{X}_T\}$.*

Then, in the nominal setting, the origin is asymptotically stable with domain of attraction $\mathcal{X}$. In the robust setting, $\mathcal{X}_T$ is input-to-state stable with domain of attraction $\mathcal{X}$.

Proof. It suffices to show that these conditions immediately imply satisfaction of the necessary assumptions in [66], together with the additional sufficient conditions of Theorem 4.2 in [67]. These results state that if

(i) f and f_T are continuous and positive definite,

(ii) $\mathcal{X}$, $\mathcal{U}$, $\mathcal{X}_T$ contain the origin and are closed,

(iii) $\mathcal{X}_T$ is control invariant, and

(iv) $\min_{u\in\mathcal{U}} f(x,u) + f_T(Ax+Bu) - f_T(x) \leq 0 \ \forall x \in \mathcal{X}_T$,

then the desired stability guarantees hold.

Condition *1)* implies satisfaction of the part of (i) that concerns f. Condition *2)* implies satisfaction of (ii). If $\mathcal{P}$ contains the origin and is closed, by definition this implies that $\mathcal{X}$ and $\mathcal{U}$ also contain the origin and are closed. Also, since $\mathcal{X}_T$ is defined in terms of $\mathcal{P}$ in (3.2), $\mathcal{X}_T$ also contains the origin and is closed. Condition *3)* implies satisfaction of (iii) by virtue of Lemma 4. Condition *4)* implies that f_T is a Lyapunov function on $\{x \in \mathbb{R}^n : \ 1 \leq f_T(x)\} \supseteq \mathcal{X}_T$ since it is the Minkowski functional of the terminal set $\mathcal{X}_T$ (see Theorem 3.3. in [65]). Therefore, the condition stated in (iv) is automatically satisfied for all $x \in \mathcal{X}_T$. Moreover, f_T is necessarily positive definite, so the part of (i) that concerns f_T is satisfied as well.

Therefore, by virtue of the results in [66] and [67], we guarantee asymptotic stability of the origin in the nominal setting and ISS of $\mathcal{X}_T$ in the robust setting, both with domain of attraction $\mathcal{X}$. □

For any cost f and constraint $\mathcal{P}$ that satisfy conditions *1)* and *2)* of Lemma 5, we can choose an appropriate terminal set $\mathcal{X}_T$ and terminal cost f_T as per *3)* and *4)* to satisfy the lemma. This guarantees stability for the DLMPC problem (3.1). However, as stated, f_T does not satisfy Assumption 1, and therefore it is not localized. In particular, f_T can be written as:

$$f_T(x) = \inf_{\eta} \{\eta \geq 0 : \ [x]_{\mathbf{in}_i(d)} \in \eta \mathcal{X}_T^i \ \forall i\}, \tag{3.4}$$

which cannot be written as a sum of local functions. Nonetheless, this scalar objective function admits a distributed and localized implementation–we can add it in the DLMPC algorithm using the ADMM-based consensus technique described in §3.4.

Convergence guarantees

Algorithm 1 relies on ADMM. We can guarantee convergence of the overall algorithm by leveraging the ADMM convergence result from [52].

Lemma 6. *In Algorithm 1, the residue, objective function, and dual variable converge as $k \rightarrow \infty$, i.e.,*

$$\tilde{H}\tilde{\mathbf{\Phi}}^k - \tilde{\mathbf{\Psi}}^k \rightarrow 0, \ f(\tilde{\mathbf{\Phi}}^k x_0) \rightarrow f(\tilde{\mathbf{\Phi}}^* x_0), \ \tilde{\mathbf{\Psi}}^k \rightarrow \tilde{\mathbf{\Psi}}^*,$$

where $\star$ indicates optimal value.

Proof. Algorithm 1 is the result of applying ADMM to the DLMPC problem (2.12), then exploiting the separability and structure of resulting sub-problems to achieve distributed and localized implementation, as presented in Chapter 2. Thus, to prove convergence, we only need to show that the underlying ADMM algorithm converges. By the ADMM convergence result in [52], the desired convergence of the residue, objective function, and dual variable are guaranteed if, for this algorithm,

1. *The extended-real-value functional is closed, proper, and convex, and*

2. *The unaugmented Lagrangian has a saddle point.*

We first show *1)*. The extended-real-value functional $h(\tilde{\mathbf{\Phi}})$ is defined for this algorithm as

$$h(\tilde{\mathbf{\Phi}}) = \begin{cases} f(M_1\tilde{\mathbf{\Phi}}\{1\}x_0) & \text{if } Z_{AB}M_2\tilde{H}^{\dagger}\tilde{\mathbf{\Phi}} = I, \\ & \tilde{\mathbf{\Phi}} \in \mathcal{L}_d, \tilde{\mathbf{\Phi}}x_0 \in \mathcal{P}, \\ \infty & \text{otherwise,} \end{cases}$$

where $\tilde{H}^{\dagger}$ is the left inverse of $\tilde{H}$ from (2.11); $\tilde{H}$ has full column rank. When formulating the DLMPC problem (3.1) as an ADMM problem, we perform variable duplication to obtain problem (2.11). We can write (2.11) in terms of $h(\tilde{\mathbf{\Phi}})$ with the constraint $\tilde{\mathbf{\Phi}} = \tilde{H}\tilde{\mathbf{\Psi}}$:

$$\min_{\tilde{\mathbf{\Phi}},\tilde{\mathbf{\Psi}}} \quad h(\tilde{\mathbf{\Phi}}) \text{ s.t. } \quad \tilde{\mathbf{\Phi}} = \tilde{H}\tilde{\mathbf{\Psi}}.$$

By assumption, $f(M_1\tilde{\mathbf{\Phi}}x_0)$ is closed, proper, and convex, and $\tilde{\mathcal{P}}$ is a closed and convex set. The remaining constraints $Z_{AB}\tilde{\mathbf{\Phi}} = I$ and $\tilde{\mathbf{\Phi}} \in \mathcal{L}_d$ are also closed and convex. Hence, $h(\tilde{\mathbf{\Phi}})$ is closed, proper, and convex.

We now show *2)*. This condition is equivalent to showing that strong duality holds [68]. Since problem (3.1) is assumed to have a feasible solution in the relative interior of $\mathcal{P}$ by means of Lemma 4 (given that the first iteration is feasible), Slater's condition is automatically satisfied, and therefore the unaugmented Lagrangian of the problem has a saddle point.

Both conditions of the ADMM convergence result from [52] are satisfied–Algorithm 1 converges in residue, objective function, and dual variable. □

Remark 6. *It is important to note that the communication network might be a directed graph–depending on the elements in the incoming as outgoing sets. The proof outlined here holds as well for the case where the communication graph is directed. For additional details on ADMM convergence over directed networks readers are referred to [69] and references therein.*

3.4 Feasible and Stable DLMPC

We incorporate theoretical results from §3.3 into the DLMPC computation. We provide an algorithm to compute the terminal set $\mathcal{X}_T$. We also provide a distributed and localized computation for the terminal cost function f_T. The terminal set and cost generally introduce local coupling among subsystems; this minimizes conservatism but requires an extension to Algorithm 1 to accommodate coupling. All algorithms provided are distributed and localized, with computational complexity that is independent of the global system size.

Offline synthesis of the terminal set $\mathcal{X}_T$

Our first result is to provide an offline algorithm to compute terminal set $\mathcal{X}_T$ from (3.2) in a distributed and localized manner. As discussed in §3.3, we compute the maximal robust positive invariant set for the unconstrained localized closed-loop system. We use SLS-based techniques to obtain a localized closed-loop map $\mathbf{\Phi}$.

Our algorithm is based on Algorithm 10.4 from [1]. The advantage of implementing this algorithm in terms of the localized closed-loop map $\mathbf{\Phi}$ is twofold: i) we can work with locally-bounded and polytopic disturbance sets $\mathcal{W}$ by leveraging Lemmas 1 and 2, and ii) the resulting robust invariant set is automatically localized given the localized structure of the closed-loop map.

We start by finding a localized closed-loop map $\mathbf{\Phi}$ for system (2.1). To do this, we need to solve

$$\min_{\mathbf{\Phi}} f(\mathbf{\Phi}) \quad \text{s.t. } Z_{AB}\mathbf{\Phi} = I,\ \mathbf{\Phi} \in \mathcal{L}_d. \tag{3.5}$$

This is an SLS problem with a separable structure for most standard cost functions f [51]. The separable structure admits localized and distributed computation. Even when separability is not apparent, a relaxation can often be found that allows for distributed computation (see for example [70]). The infinite-horizon solution for quadratic cost is presented in [71]. For other costs, computing an infinite horizon solution to (3.5) remains an open question–in these cases, we can use finite impulse response SLS with sufficiently long time horizon.

Once a localized closed-map $\mathbf{\Phi}$ has been found, we can compute the associated maximal positive invariant set. In the robust case, we leverage results from Lemmas 1 and 2, which use duality arguments to tackle specific formulations of $\mathcal{W}$. We denote σ as the upper bound of $\|[\delta]_i\|_*$ for all i, where $\|\cdot\|_*$ is the dual norm of $\|\cdot\|_p$. Also, Each e_j is the j^{th} vector in the standard basis.

- Nominal (i.e., no disturbance):

$$\mathcal{S} := \{x_0 \in \mathbb{R}^n : \ \mathbf{\Phi}\{1\}x_0 \in \mathcal{P}\}.$$

- Locally bounded disturbance:

$$\mathcal{S} := \{x_0 \in \mathbb{R}^n : \ [H]_i[\mathbf{\Phi}\{1\}]_i[x_0]_i + \sum_j \sigma \left\| e_j^\top [H]_i[\mathbf{\Phi}\{2:T\}]_i \right\|_* \leq [h]_i \ \forall i\}.$$

- Polytopic disturbance:

$$\mathcal{S} := \{x_0 \in \mathbb{R}^n : \ H\mathbf{\Phi}\{1\}x_0 + \Xi g \leq h, \text{where}$$
$$\Xi(j,:) = \min_{\Xi_j \geq 0} \Xi_j g \text{ s.t. } H(j,:)\mathbf{\Phi}\{2:T\} = \Xi_j G \ \forall j\}.$$

As per Lemma 4, we calculate $\mathcal{S}$ by iteratively computing $\mathcal{S}_{k+1}$ from $\mathcal{S}_k$.[1] Assume $\mathcal{S}_k$ can be written as

$$\mathcal{S}_k = \{x_0 \in \mathbb{R}^n : \ \hat{H}x \leq \hat{h}\}.$$

Then, we can write $\mathcal{S}_{k+1}$ as

$$\mathcal{S}_{k+1} = \{x_0 \in \mathbb{R}^n : \ \Phi_{x,1}[1]x_0 \in \mathcal{F}(\mathcal{S}_k), \Phi_{u,0}[0]x_0 \in \mathcal{U}\}, \tag{3.6}$$

where $\mathcal{F}(\mathcal{S}_k) := \mathcal{S}_k$ in the nominal case. In the case of locally bounded disturbance,

$$\mathcal{F}(\mathcal{S}_k) := \{x \in \mathbb{R}^n : \ [\hat{H}]_i[x]_i + \sum_j \sigma \left\| e_j^\top [\hat{H}]_i \right\|_* \leq [\hat{h}]_i \ \forall i\},$$

and for polytopic disturbance,

$$\mathcal{F}(\mathcal{S}_k) := \{x \in \mathbb{R}^n : \ \hat{H}x + \Xi g \leq \hat{h},$$

where $\Xi(j,:) = \min_{\Xi_j \geq 0} \Xi_j g$ s.t. $\hat{H}(j,:) = \Xi_j G \ \forall j\}$.

These formulations use the simplifying fact that $\Phi_{x,0}[0] = I$ for all feasible closed-loop dynamics. Also, notice that by using $\mathcal{S}_k$ to calculate $\mathcal{S}_{k+1}$, the only elements of $\mathbf{\Phi}$ that we require are $\Phi_{x,1}[1]$ and $\Phi_{u,0}[0]$.

The conditions stated in (3.6) are row- and column-wise separable in $\mathbf{\Phi}$ (and Ξ); this allows us to calculate $\mathcal{S}_k$ in a distributed way. Furthermore, $\mathcal{F}(\mathcal{S}_k)$ and $\mathcal{U}$ are localizable by Assumption 1, in both nominal and robust settings. Since $\mathbf{\Phi}$ is also localized, we can exploit the structure of $\mathbf{\Phi}$ to and rewrite S_{k+1} as the intersection of local sets, i.e., $S_{k+1} = S_{k+1}^1 \cap \cdots \cap S_{k+1}^N$, where

$$\begin{aligned} \mathcal{S}_{k+1}^i = \{[x_0]_{\mathbf{in}_i(d)} \in \mathbb{R}^{[n]_i} : & \\ & [\Phi_{x,1}[1]]_{\mathbf{in}_i(d)}[x_0]_{\mathbf{in}_i(d)} \in \mathcal{F}(\mathcal{S}_k^{\mathbf{in}_i(d)}), \\ & [\Phi_{x,1}[1]]_{\mathbf{in}_i(d)}[x_0]_{\mathbf{in}_i(d)} \in \mathcal{U}^{\mathbf{in}_i(d)}\}. \end{aligned} \tag{3.7}$$

We now present Algorithm 2 to compute the terminal set $\mathcal{X}_T := \mathcal{S}$ in a distributed and local manner. Each subsystem i computes its own local terminal set $\mathcal{S}^i$ using only local information exchange. This algorithm is inspired by Algorithm 10.4 in [1].

[1]For simplicity of presentation, we write out $\mathcal{S}$ only for finite-horizon $\mathbf{\Phi}$; the proposed algorithm to synthesize $\mathcal{S}$ works for infinite-horizon $\mathbf{\Phi}$ as well.

Algorithm 2 Subsystem i terminal set computation

```
   input: $[\Phi_{x,1}[1]]_{\mathbf{in}_i(d)}$, $[\Phi_{u,0}[0]]_{\mathbf{in}_i(d)}$, $\mathcal{U}^{\mathbf{in}_i(d)}$
       for polytopic noise, also $[G]_{\mathbf{in}_i(d)}$, $[g]_{\mathbf{in}_i(d)}$
1: $\mathcal{S}_0^i \leftarrow \mathcal{X}^i, k \leftarrow -1$
2: repeat:
3:     $k \leftarrow k+1$
4:     Share $\mathcal{S}_k^i$ with $\mathbf{out}_i(d)$ .
       Receive $\mathcal{S}_k^j$ from all $j \in \mathbf{in}_i(d)$.
5:     Compute $\tilde{\mathcal{S}}_{k+1}^i$ via (3.7).
6:     $\mathcal{S}_{k+1}^i \leftarrow \mathcal{S}_k^i \cap \tilde{\mathcal{S}}_{k+1}^i$
7: until: $\mathcal{S}_{k+1}^i = \mathcal{S}_k^i$ for all $i$
8: $\mathcal{X}_T^i \leftarrow \mathcal{S}_k^i$
   output: $\mathcal{X}_T^i$
```

If state and input constraints $\mathcal{X}$ and $\mathcal{U}$ induce no coupling (as assumed in Algorithm 1), the resulting terminal set $\mathcal{X}_T$ will be d-localized. If $\mathcal{X}$ and $\mathcal{U}$ induce d-local coupling, the terminal set will be $2d$-localized since–in the presence of coupling–Alg. 2 requires communication with not only local patch neighbors, but also neighbors of those neighbors. Convergence is guaranteed for system (2.1) if it is stable when $u = 0,\ w = 0$, and when the constraint and disturbance sets $\mathcal{X},\ \mathcal{U},\ \mathcal{W}$ are bounded and contain the origin, as per [72].

Computational complexity of the algorithm: In Algorithm 2, step 4 refers to communication exchanges, while steps 5, 6, and 7 tackle computational steps. In terms of the computational complexity of Algorithm 2, the order of magnitude of the computations at each local subsystem i is $O(m_i d)$, where m_i is the number of constraints on subsystem i. Notice that in the case of a coupled scenario, the number of constraints at subsystem i, m_i, is in general larger than in the uncoupled case, since additional constraints couple neighboring subsystems. Notice also that the communication exchanges occur only within each d-local neighborhood, and therefore the computation of the maximal robust positive invariant set can be computed in a purely distributed and localized manner. Hence, its scalability is independent from the size of the whole system. It is also worth noting that this computation occurs offline, where both communication and computational complexity are not so critical.

Online synthesis of DLMPC

DLMPC Algorithm 1 was developed under the assumption that no coupling is introduced through cost or constraints. We now extend this algorithm to accommodate local coupling, which is allowed as per Assumption 1. This allows us to incorporate the terminal set and cost introduced in the previous sections, both of which generally induce local coupling. We will use notation that assumes all constraints and costs (including the terminal set) are d-localized. In the case that the terminal set is $2d$-localized, subsystems will need to exchange information with neighbors up to $2d$-hops away; simply replace $\{\ \mathbf{out}_i(d), \mathbf{in}_i(d)\ \}$ with $\{\ \mathbf{out}_i(2d), \mathbf{in}_i(2d)\ \}$ wherever they appear in Algorithms 1, 2, and 3.

Appending the terminal set (3.2) and terminal cost (3.4) to the DLMPC problem (2.11) gives:

$$\begin{aligned} \min_{\tilde{\mathbf{\Phi}}, \tilde{\mathbf{\Psi}}, \eta} \quad & f(M_1 \tilde{\mathbf{\Phi}}\{1\} x_0) + \eta \\ \text{s.t.} \quad & Z_{AB} M_2 \tilde{\mathbf{\Psi}} = I, x_0 = x(\tau),\ \tilde{\mathbf{\Phi}}, \tilde{\mathbf{\Psi}} \in \mathcal{L}_d, \\ & \tilde{\mathbf{\Phi}} \in \tilde{\mathcal{P}},\ \tilde{H} \tilde{\mathbf{\Phi}} = \tilde{\mathbf{\Psi}}, \\ & M_1 \tilde{\mathbf{\Phi}}_T \{1\} x_0 \in \eta \mathcal{X}_T, 0 \le \eta \le 1, \end{aligned} \tag{3.8}$$

where $\tilde{\mathbf{\Phi}}_T$ represents block rows of $\tilde{\mathbf{\Phi}}$ that correspond to time horizon T–e.g., in the nominal setting, the last block row of $\mathbf{\Phi}_x$, and the set $\tilde{\mathcal{P}}$ is defined so that translates the constraint on $\tilde{\mathbf{\Phi}} x_0 \in \mathcal{P}$ into a constraint directly on $\tilde{\mathbf{\Phi}}$.

Remark 7. *In the robust setting, we incorporate the terminal constraint into $\tilde{\mathcal{P}}$ to ensure robust satisfaction of the terminal constraint. However, $\mathcal{X}_T$ still appears as nominal constraint in order to integrate the definition of the terminal cost* (3.4) *into the DLMPC formulation* (3.1).

In the original formulation (2.11), we assume no coupling; as a result, all expressions involving $\tilde{\mathbf{\Phi}}$ are row-separable, and all expressions involving $\tilde{\mathbf{\Psi}}$ are column-separable. If we allow local coupling as per Assumption 1, we lose row-separability; row-separability is also lost in the new formulation (3.8) due to local coupling in the terminal set. This is because without coupling, $[x]_i$ and $[u]_i$ (which correspond to a fixed set of rows in $\mathbf{\Phi}$), can only appear in f^i and $\mathcal{P}^i$; thus, each row of $\mathbf{\Phi}$ (and $\tilde{\mathbf{\Phi}}$) is solved by exactly one subsystem in step 3 of Algorithm 1. With coupling, $[x]_i$ and $[u]_i$ can appear in f^j and $\mathcal{P}^j$ for any $j \in \mathbf{in}_i(d)$; now, each row of $\mathbf{\Phi}$ is solved for by multiple local subsystems at once, and row-separability is lost. This only affects step 3 of Algorithm 1 (i.e., the row-wise problem); other steps remain unchanged.

We write out the row-wise problem corresponding to (3.8):

$$\begin{aligned} \min_{\tilde{\mathbf{\Phi}}} \quad & f(M_1\tilde{\mathbf{\Phi}}\{1\}x_0) + \eta + g(\tilde{\mathbf{\Phi}}, \tilde{\mathbf{\Psi}}^k, \mathbf{\Lambda}^k) \\ \text{s.t.} \quad & \tilde{\mathbf{\Phi}} \in \tilde{\mathcal{P}} \cap \mathcal{L}_d,\ M_1\tilde{\mathbf{\Phi}}_T\{1\}x_0 \in \eta\mathcal{X}_T, \\ & 0 \le \eta \le 1,\ x_0 = x(\tau), \end{aligned} \tag{3.9}$$

where $g(\tilde{\mathbf{\Phi}}, \tilde{\mathbf{\Psi}}, \mathbf{\Lambda}) = \frac{\rho}{2}\left\|\tilde{\mathbf{\Phi}} - \tilde{H}\tilde{\mathbf{\Psi}} + \mathbf{\Lambda}\right\|_F^2$, and $\mathbf{\Lambda}$ is the additional variable introduced by ADMM.

Note that if $\mathcal{P}$ induces coupled linear inequality constraints, row-separability is maintained. Though $[x]_i$ and $[u]_i$ appear in multiple $\mathcal{P}_j$, this corresponds to distinct rows of $\mathbf{\Omega}$, each solved by one subsystem, as opposed to one row of $\mathbf{\Phi}$ that is solved for by multiple subsystems. A similar idea applies if coupling is induced by some quadratic cost, i.e., $f(\mathbf{\Phi}x_0) = \|C\mathbf{\Phi}x_0\|_F^2$ for some matrix C which induces coupling. In this case, we can modify $\tilde{H}$ such that we enforce $\mathbf{\Phi} = C\mathbf{\Psi}$, and rewrite the cost as $\|\mathbf{\Phi}x_0\|_F^2$; now, each row of $\mathbf{\Phi}$ is solved by exactly one subsystem, and row-separability is maintained.[2] However, this technique does not apply to coupling in the general case; nor does it apply to the coupling induced by the terminal cost–it is generally true that each row of $\tilde{\mathbf{\Phi}}$ will be need to be solved for by several subsystems. We introduce a new vector variable

$$\mathbf{X} := M_1\tilde{\mathbf{\Phi}}\{1\}x_0.$$

This variable facilitates consensus between subsystems who share the same row(s) of $\tilde{\mathbf{\Phi}}$. Each subsystem solves for components $[\mathbf{X}]_{\mathbf{in}_i(d)}$, and comes to a consensus with its neighboring subsystems on the value of these components. In the interest of efficiency, we directly enforce consensus on elements of $\mathbf{X}$ instead of enforcing consensus on rows of $\tilde{\mathbf{\Phi}}$. We introduce a similar variable for terminal cost η; we define vector $\boldsymbol{\eta} := \begin{bmatrix}\eta_1, \dots, \eta_N\end{bmatrix}$. Subsystem i solves for η_i, which is its own copy of η. It then comes to a consensus on this value with its neighbors, i.e., η_j has the same value $\forall j \in \mathbf{in}_i(d)$. Assuming that $\mathcal{G}_{(A,B)}$ is connected, this guarantees that η_i has the same value $\forall i \in \{1 \dots N\}$, i.e., all subsystems agree on the value of η. We combine these two consensus-facilitating variables into the augmented vector variable

$$\tilde{\mathbf{X}} := \begin{bmatrix}\mathbf{X}^\intercal & \boldsymbol{\eta}^\intercal\end{bmatrix}^\intercal.$$

With this setup, we follow a variable duplication strategy and apply ADMM for consensus [73]. In particular, we duplicate variables so each subsystem has *its own*

[2]We would also need to use $[\Psi_{u,0}[0]]_i$ for the algorithm output instead of $[\Phi_{u,0}[0]]_i$.

copy of the components of $\tilde{\mathbf{X}}$. Problem (3.9) becomes:

$$\begin{aligned} \min_{\tilde{\mathbf{\Phi}},\tilde{\mathbf{X}},\tilde{\mathbf{Y}}} \quad & \tilde{f}(\tilde{\mathbf{X}}) + g(\tilde{\mathbf{\Phi}}, \tilde{\mathbf{\Psi}}^k, \mathbf{\Lambda}^k) \\ \text{s.t.} \quad & \tilde{\mathbf{\Phi}}, \tilde{\mathbf{X}} \in \mathcal{Q},\ \tilde{\mathbf{X}} \in \tilde{\mathcal{X}}_T,\ \tilde{\mathbf{\Phi}} \in \mathcal{L}_d,\ x_0 = x(\tau), \\ & [\tilde{\mathbf{X}}]_i = [M_1 \tilde{\mathbf{\Phi}}]_{i_r} [x_0]_i,\ [\tilde{\mathbf{X}}]_j = [\tilde{\mathbf{Y}}]_j\ \forall j \in \mathbf{in}_i(d)\ \forall i \end{aligned} \tag{3.10}$$

where we define $\tilde{f}$, $\tilde{\mathcal{X}}_T$, and $\mathcal{Q}$ as follows:

- $\tilde{f}(\tilde{\mathbf{X}}) := f(\mathbf{X}) + \frac{1}{N}\sum_{i=1}^{N} \eta_i$,
- $\tilde{\mathbf{X}} \in \tilde{\mathcal{X}}_T$ if and only if $M_1 \tilde{\mathbf{\Phi}}_T\{1\} x_0 \in \eta_i \mathcal{X}_T\ \forall i$,
- $\tilde{\mathbf{\Phi}}, \tilde{\mathbf{X}} \in \mathcal{Q}$ if and only if $\mathbf{0} \le \boldsymbol{\eta} \le \mathbf{1}$ **and** $\{\tilde{\mathbf{\Phi}} \in \tilde{\mathcal{P}}$ if $\tilde{\mathcal{P}}$ induces linear inequalities **or** $\tilde{\mathbf{X}} \in \mathcal{P}$ otherwise$\}$.[3]

The structure of problem (3.10) allows us to solve it in a distributed and localized manner via ADMM-based consensus. In particular, each subsystem i solves:

$$\begin{Bmatrix} [\tilde{\mathbf{\Phi}}]_{i_r,}^{k+1,n+1} \\ \\ [\tilde{\mathbf{X}}]_{\mathbf{in}_i(d)}^{n+1} \end{Bmatrix} = \begin{Bmatrix} \underset{[\tilde{\mathbf{\Phi}}]_{i_r}, [\tilde{\mathbf{X}}]_{\mathbf{in}_i(d)}}{\operatorname{argmin}} & \tilde{g}_k^i(\tilde{\mathbf{X}}, \tilde{\mathbf{\Phi}}) + \frac{\mu}{2} h^i(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}^n, \tilde{\mathbf{Z}}^n) \\ \text{s.t.} & [\tilde{\mathbf{\Phi}}]_{r_i} \in \mathcal{Q}^i,\ [\tilde{\mathbf{X}}_T]_i \in \tilde{\mathcal{X}}_T^i \cap \mathcal{Q}^i, \\ & [\tilde{\mathbf{X}}]_i = [M_1 \tilde{\mathbf{\Phi}}]_{r_i} [x_0]_i \end{Bmatrix} \tag{3.11a}$$

$$[\tilde{\mathbf{Y}}]_i^{n+1} = \frac{h^i(\tilde{\mathbf{X}}^{n+1}, \mathbf{0}, \tilde{\mathbf{Z}}^n)}{|\mathbf{in}_i(d)|}, \tag{3.11b}$$

$$[\tilde{\mathbf{Z}}]_{ij}^{n+1} = [\tilde{\mathbf{Z}}]_{ij}^{n} + [\tilde{\mathbf{X}}]_i^{n+1} - [\tilde{\mathbf{Y}}]_j^{n+1}, \tag{3.11c}$$

where to simplify notation, we define

$$\begin{aligned} \tilde{g}_k^i(\tilde{\mathbf{X}}, \tilde{\mathbf{\Phi}}) &:= \tilde{f}^i([\tilde{\mathbf{X}}]_{\mathbf{in}_i(d)}) + g\big([\mathbf{\Phi}]_{i_r} - [\mathbf{\Psi}]_{i_r}^k + [\mathbf{\Lambda}]_{i_r}^k\big), \\ h^i(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}^n, \tilde{\mathbf{Z}}^n) &:= \sum\nolimits_{j \in \mathbf{in}_i(d)} \left\| [\mathbf{X}]_j - [\mathbf{Y}]_i^n + [\mathbf{Z}]_{ij}^n \right\|_F^2 . \end{aligned}$$

Consensus iterations are denoted by n, outer-loop (i.e., Algorithm 1) iterations are denoted by k, and μ is the ADMM consensus parameter. Intuitively, $\tilde{g}_k^i$ represents the original objective from (3.9), and h^i represents the consensus objective.

The subroutine described by (3.11) allows us to accommodate local coupling induced by cost and constraint (including terminal), and can be implemented in a distributed

[3]By the assumptions in [53], $\mathcal{P}$ (and therefore $\tilde{\mathcal{P}}$) must induce linear inequalities in the robust setting. The only case where $\tilde{\mathcal{P}}$ may induce something other than linear inequalities is in the nominal setting.

and localized manner. As stated above, this subroutine solves the row-wise problem (3.9), corresponding to step 3 of Algorithm 1. Thus, in order to accommodate local coupling (including terminal set and cost), we need only to replace step 3 of Algorithm 1 with the subroutine defined by Algorithm 3 below. Convergence is guaranteed by a similar argument to Lemma 6.

Algorithm 3 Subsystem i implementation of step 3 in Algorithm 1 when subject to localized coupling

```
    input: tolerance parameters $\epsilon_x, \epsilon_z, \mu > 0$.
1:  $n \leftarrow 0$.
2:  Solve optimization problem (3.11a).
3:  Share $[\mathbf{X}]_i^{n+1}$ with $\mathbf{out}_i(d)$. Receive the corresponding $[\mathbf{X}]_j^{n+1}$ from $j \in \mathbf{in}_i(d)$.
4:  Perform update (3.11b).
5:  Share $[\mathbf{Y}]_i^{n+1}$ with $\mathbf{out}_i(d)$. Receive the corresponding $[\mathbf{Y}]_j^{n+1}$ from $j \in \mathbf{in}_i(d)$.
6:  Perform update (3.11c).
7:  if $\left\|[\mathbf{X}]_i^{n+1} - [\mathbf{Z}]_i^{n+1}\right\|_F < \epsilon_x$
    and $\left\|[\mathbf{Z}]_i^{n+1} - [\mathbf{Z}]_i^{n}\right\|_F < \epsilon_z$ :
      Go to step 4 in Algorithm 1.
    else: Set $n \leftarrow n + 1$ and return to step 2.
```

Computational complexity of the algorithm: The presence of coupling induces an increase in computational complexity compared to the uncoupled scenario (i.e., Algorithm 1); however, the scalability properties from the uncoupled scenario still apply. Complexity in the current algorithm is determined by steps 2, 4, 6 of Algorithm 3 and steps 5 and 7 of Algorithm 1. Except for step 2 in Algorithm 3, all other steps can be solved in closed form. Sub-problems solved in step 2 require $O(d^2T^2)$ optimization variables in the robust setting ($O(dT^2)$ in the nominal setting) and $O(d^2T)$ constraints. All other steps enjoy less complexity since their evaluation reduces to multiplication of matrices of dimension $O(d^2T^2)$ in the robust setting, and $O(d^2T)$ in the nominal setting. The difference in complexity between the nominal and robust settings is consistent with the uncoupled scenario. Compared to the uncoupled scenario, additional computation burden is incurred by the consensus subroutine in Algorithm 3, which increases the total number of iterations. The consensus subroutine also induces increased communication between subsystems, as it requires local information exchange. However, this exchange is limited to a d-local region, resulting in small consensus problems that converge quickly, as we illustrate empirically in §3.5. As with the original uncoupled algorithm, complexity of this new algorithm is determined by the size of the local neighborhood and does not increase with the size of the global network.

3.5 Simulation Experiments

Using examples, we demonstrate how adding terminal constraint and cost affects the performance of the DLMPC algorithm. We verify that introducing terminal constraint and cost produces the desired feasibility and stability. We empirically characterize the computational complexity of algorithms presented in previous sections and verify that complexity is independent of global network size for both offline and online algorithms.[4]

System model

Simulations are performed on the same system as in Chapter 2; a two-dimensional square mesh, where each node represents a two-state subsystem that follows linearized and discretized swing dynamics

$$\begin{bmatrix}\theta(t+1)\\ \omega(t+1)\end{bmatrix}_i = \sum_{j\in\mathbf{in}_i(1)} [A]_{ij}\begin{bmatrix}\theta(t)\\ \omega(t)\end{bmatrix}_j + [B]_i[u]_i + [w]_i,$$

where $[\theta]_i$, $[\dot{\theta}]_i$, $[u]_i$ are the phase angle deviation, frequency deviation, and control action of the controllable load of bus i. The dynamic matrices are

$$[A]_{ii} = \begin{bmatrix}1 & \Delta t\\ -\frac{k_i}{m_i}\Delta t & 1-\frac{d_i}{m_i}\Delta t\end{bmatrix},\ [A]_{ij} = \begin{bmatrix}0 & 0\\ \frac{k_{ij}}{m_i}\Delta t & 0\end{bmatrix},\ \text{and } [B]_{ii} = \begin{bmatrix}0 & 1\end{bmatrix}^\mathsf{T} \text{ for all } i.$$

Connectivity among nodes is determined at random; each node connects to each of its neighbors with a 40% probability. The expected number of edges is $0.8*n*(n-1)$. The parameters in bus i: m_i^{-1} (inertia inverse), d_i (damping) and k_{ij} (coupling term) are randomly generated and uniformly distributed between [0, 2], [0.5, 1], and [1, 1.5], respectively. We set the discretization step $\Delta t = 0.2$, and define $k_i := \sum_{j\in\mathbf{in}_i(1)} k_{ij}$.

We study both the nominal setting and robust setting with uniformly distributed polytopic noise. The baseline parameter values are $d = 3$, $T = 5$, $N = 16$ (4×4 grid). Unless otherwise specified, we start with a random-generated initial condition. We use a quadratic cost and polytopic constraints on both angle and frequency deviation, and impose upper and lower bounds.

[4]Code to replicate these experiments is available at `https://github.com/unstable-zeros/dl-mpc-sls`; this code makes use of the SLS toolbox [59] at `https://github.com/sls-caltech/sls-code`, which includes ready-to-use MATLAB implementations of all algorithms presented in this thesis.

Performance

The addition of the terminal set and cost to the DLMPC algorithm introduces minimal conservatism in both nominal and robust settings. We study the the DLMPC cost for varying values of the time horizons for three different cases: (i) without terminal set and cost (ii) with terminal set, (iii) with terminal set *and* terminal cost. Results are summarized in Fig. 3.1. Observe that the difference between the optimal cost across all three cases are negligible, indicating that our proposed terminal set and cost introduce no conservatism (while still providing the necessary theoretical guarantees).

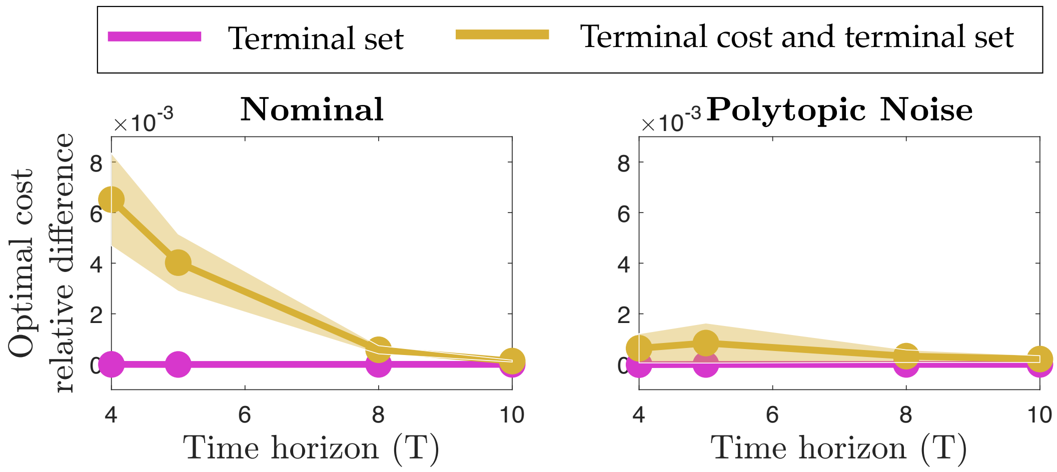

Figure 3.1: Relative difference of the optimal cost obtained (i) with terminal set (pink) and (ii) with both terminal set and cost (yellow) compared to the optimal cost computed without terminal set and cost. Relative difference is obtained by taking the difference of the two costs and normalizing by the non-terminal-constrained cost. The difference obtained by adding the terminal set (indicated in pink) is on the order of 10^{-5} and is not visible on the plot.

Generally, the inclusion of the terminal set and cost introduce minimal change. In the vast majority of cases (e.g., all simulations from Chapter 2), the DLMPC algorithm is feasible and stable even without a terminal set. This phenomenon has already been observed for the centralized case [1]. However, we also want to demonstrate how the terminal set and cost *can* make a difference–example subsystem trajectories for the three different cases are shown in Fig. 3.2 for the nominal case and in the presence of polytopic disturbances. For these simulations only, we use a smaller ($N = 5$), more unstable (m_i^{-1} between $[0, 16]$) system, extremely short time horizon ($T = 2$), and somewhat hand-crafted initial states and disturbances to obtain clearly visible differences between cases–without such instability, short time horizon, and hand-crafting, differences are generally tiny and not visible. In all cases, the centralized solution (computed via CVX [60]) coincides with the solution achieved by the DLMPC Algorithm 1, validating the optimality of the proposed algorithm. The effects of introducing terminal set and terminal cost are apparent and consistent with the theoretical results presented in this paper.

Computational complexity

Simulation results verify the scalability of the proposed methods. We measure runtime[5] while varying different network and problem parameters: locality d, network size N, and time horizon T.[6] We run 5 different simulations for each of the parameter combinations, using different realizations of the randomly chosen parameters to provide consistent runtime estimates.

First, we study the scalability of the offline Algorithm 2 to compute the terminal set; results are shown in Fig. 3.3. Consistent with theoretical analyses in §3.4, runtime does not increase with the size of the network; rather, it increases with the size of the neighborhood. As expected, computations for the robust set take slightly longer than for the nominal set, since the variables in the robust setting have greater dimension. Overall, synthesis times for both nominal and robust settings are extremely low, especially when a small locality size is used.

We also study how scalability of the DLMPC algorithm is affected when we impose a terminal set and terminal cost, and use Algorithm 3 to handle coupling. Results are shown in Fig. 3.4; this figure was generated using the same systems and parameters as Fig. 6.7, allowing for direct comparison of online runtimes. The addition of the terminal set/cost slightly increases runtime, as expected. In the nominal case, runtime is increased from about 10^{-3}s to 10^{-1}s. In the case of polytopic disturbances, runtime is increased from about $1 - 10$s to 10s. Scalability is maintained; runtime barely increases with the size of the network. Overall, simulations indicate that the introduction of a terminal set and cost preserve scalability, minimally impact computational overhead and performance, and provide the desired guarantees.

[5] In online simulations, runtime is measured after the first iteration, so that all iterations for which runtime is measured are warm-started.

[6] To increase network size, we vary the size of the grid over 4×4 (32 states), 6×6 (72 states), 8×8 (128 states), and 11×11 (242 states) grid sizes.

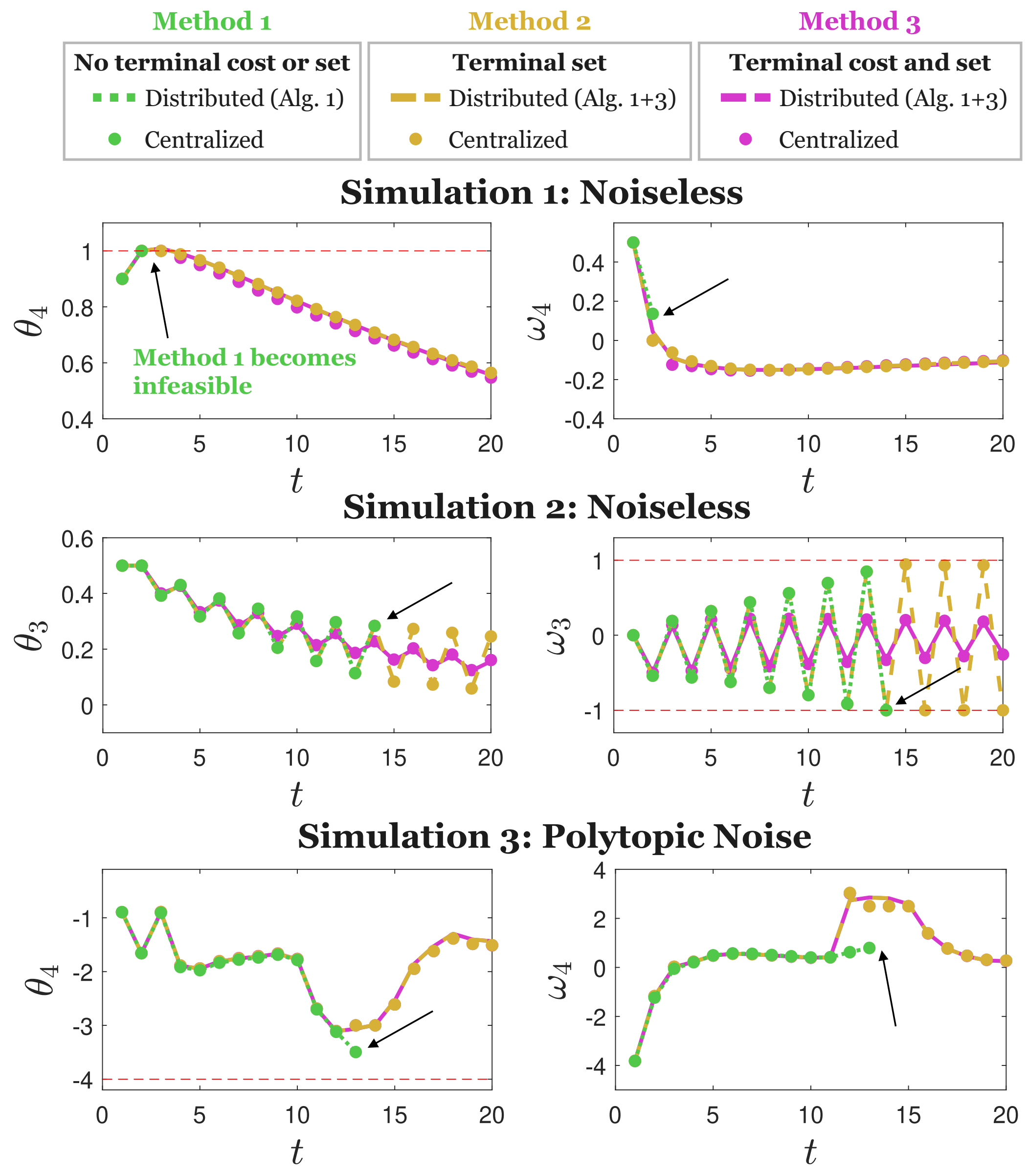

Figure 3.2: Comparison of behaviors across methods. For simulations with various initial conditions and state bounds: method 1 (green) uses DLMPC with no terminal cost or set, method 2 (yellow) uses DLMPC with only a terminal set, and method 3 (pink) uses DLMPC with both terminal set and terminal cost. In all simulations shown, method 1 becomes infeasible at some time, highlighted by the arrow; however, both methods 2 and 3 maintain feasibility, due to the inclusion of the terminal set constraint. Dashed red lines indicate state constraints. **Simulation 1**: noiseless evolution of the states of subsystem 4. Method 1 becomes infeasible after $t = 2$. Methods 2 and 3 appear very similar; the addition of the terminal cost introduces little change. **Simulation 2**: noiseless evolution of the states of subsystem 3. Method 1 becomes infeasible after $t = 14$. Here, method 2 displays large oscillations; but method 3 does not: here, the inclusion of the terminal cost plays a large role in improving performance. **Simulation 3**: evolution of the states of subsystem 4 under polytopic noise. Method 1 becomes infeasible after $t = 13$. Methods 2 and 3 appear very similar.

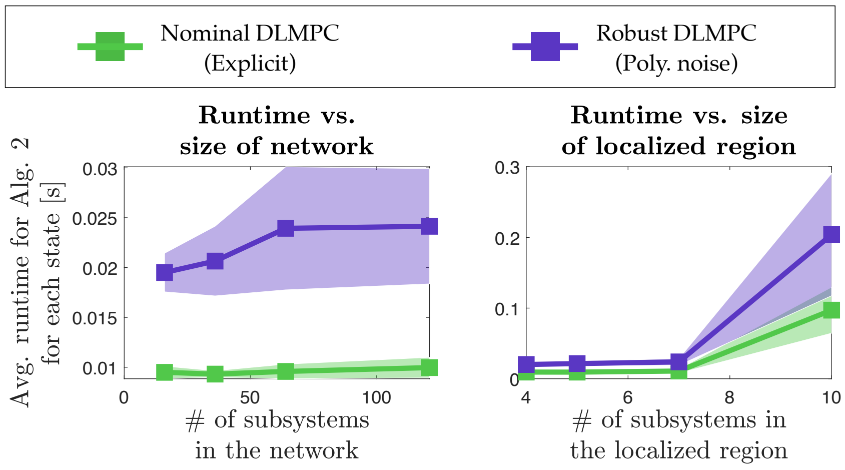

Figure 3.3: Average runtime of Algorithm 2 with network size (left) and locality parameter (right). The lines are the mean values and the shaded areas show the values within one standard deviation. Since computations are parallelized across subsystems, runtime is measured on a subsystem, normalized per state, and averaged after the algorithm computation is finished.

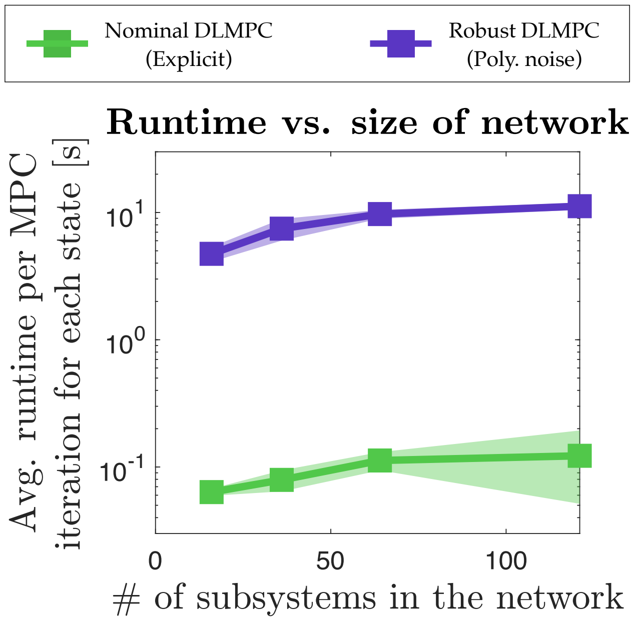

Figure 3.4: Average runtime per DLMPC iteration with network size when terminal set and terminal cost are imposed. The lines are the mean values and the shaded areas show the values within one standard deviation. Since computations are parallelized across subsystems, runtime is measured on a subsystem, normalized per state, and averaged out after the MPC controller is finished.

3.6 Conclusion

In this chapter we provide theoretical guarantees for the *closed-loop* DLMPC approach, in both nominal and robust settings. In particular, we ensure recursive feasibility and stability by incorporating a terminal set and terminal cost. We also give guarantees for convergence of the algorithm. For the terminal set, we choose the maximal robust positive invariant set, which can be expressed compactly in the SLS parametrization. We introduce an algorithm to scalably compute this terminal set. We also provide the requisite modifications to the online DLMPC algorithm to accommodate local coupling induced by the terminal set and cost, as well as general coupling induced by process cost and constraints. All algorithms require only local information exchange, and enjoy computational complexity that is independent of the global system size. Although the cost and delays of the communication exchanges are beyond the scope of this paper, if the communication delays and package dropouts can be modeled either as a polytopic or locally bounded disturbance, then the guarantees provided in this paper for convergence, recursive feasibility and asymptotic stability still hold. The results in this paper are the first to provide a distributed and localized computation of the maximal robust positive invariant control set and Lyapunov function of a large-scale system.

Chapter 4

GLOBAL PERFORMANCE GUARANTEES

Abstract

In this chapter, we provide analysis and guarantees on global performance of localized MPC–in particular, we derive sufficient conditions for optimal global performance in the presence of local communication constraints. We also present an algorithm to determine the communication structure for a given system that will preserve performance while minimizing computational complexity. The effectiveness of the algorithm is verified in simulations, and additional relationships between network properties and performance-preserving communication constraints are characterized. Overall, this chapter offers theoretical understanding on the effect of local communication on global performance, and provides practitioners with the tools necessary to deploy localized model predictive control by establishing a rigorous method of selecting local communication constraints. This work also demonstrates that the inclusion of severe communication constraints need not compromise global performance.

The content in this chapter has been published in [74].

4.1 Introduction

Distributed control is crucial for the operation of large-scale networks such as power grids and intelligent transport systems. Distributed model predictive control (MPC) is of particular interest, since MPC is one of the most powerful and commonly used control methods. The distributed and localized MPC (DLMPC) method discussed in this thesis provides scalable algorithms with stability and feasibility guarantees. As the name suggests, DLMPC incorporates local communication constraints, which are typically encapsulated in a single parameter d (rigorously defined in Section 5.3). The question of how to select this parameter remains unresolved, as two opposing forces come into play: smaller values of d represent stricter communication constraints, which correspond to decreased complexity–however, overly strict communication constraints may render the problem infeasible, or compromise system performance. In this work, we address this problem by providing a rigorous characterization of the impact of local communication constraints on performance. We focus on cases in which optimal global performance may be obtained with local communication; in other words, the performance of the system is unchanged by the introduction of communication constraints. We find that under reasonable assumptions, optimal global performance can be achieved with relatively strict local communication constraints.

Prior work

For large networked systems, several studies have been conducted on the use of offline controllers with local communication constraints. Local communication can facilitate faster computational speed [28] and convergence [29], particularly in the presence of delays [75]–however, this typically comes at the cost of supoptimal global performance [30]. In [31], a trade-off between performance and decentralization level (i.e., amount of global communication) is found for a truncated linear quadratic regulator. In system level synthesis, the offline predecessor of DLMPC [51], localization is typically associated with reduced performance of around 10% relative to the global controller. More generally, for both global and localized control, the topology of the network and actuator placement [76] plays a role in achievable controller performance [32], [33] and convergence [34]. In the realm of predictive control, communication constraints are important considerations [35]. However, the improved computational speeds offered by local predictive controllers typically come at the cost of suboptimal global performance and lack of stability and convergence guarantees [36]. The novel DLMPC method overcomes some of these drawbacks, providing both stability and convergence guarantees–however, thus far, its performance has not been substantially compared to that of the global, full-communication controller.[1] In the few instances that it has, it performed nearly identically to the global controller despite the inclusion of strict communication constraints [77], prompting further investigation.

Contributions

This work contains two key contributions. First, we provide a rigorous characterization of how local communication constraints restrict (or preserve) the set of trajectories available under predictive control, and use this to provide guarantees on optimal global performance for localized MPC. Secondly, we provide an exact method for selecting an appropriate locality parameter d for localized MPC. To the best of our knowledge, these are the first results of this kind on local communication constraints; our findings are useful to theoreticians and practitioners alike.

4.2 Problem Statement

In Chapter 2, we provided a formulation for the MPC problem (2.3). In order to solve problem (2.3) in a distributed manner while respecting local communication constraints, we introduced information exchange constraints through equation (2.4). These constraints, which we shall refer to interchangeably as *local communication constraints* or *locality constraints*, restrict sub-controllers to exchange their state and control actions with neighbors at most d-hops away, as per the incoming and outgoing sets from Definition 1. The goal of this chapter is to investigate how local communication constraints restrict the optimal trajectory provided as a solution of the MPC problem (2.3), and identify under which conditions the inclusion of locality constraints does not impact the optimal solution of the MPC problem. We also investigate a principled method to select an appropriate locality parameter d such that the optimality is preserved.

In this chapter, we restrict our analysis to the noiseless case. Notice that in the where no driving noise is present, the disturbance vector takes the form $\mathbf{w} := \begin{bmatrix} x_0^\top & 0 & \dots & 0 \end{bmatrix}^\top$. Then, only the first block column of $\mathbf{\Phi}$ needs to be computed.

[1]Prior work focuses on comparisons between centralized and distributed optimization schemes for localized MPC.

In this case, problem (2.7) at time-step τ becomes:

$$\begin{aligned} \min_{\mathbf{\Phi}\{1\}} \quad & f(\mathbf{\Phi}\{1\}x_0) && \text{(4.1a)} \\ \text{s.t.} \quad & x_0 = x(\tau), && \text{(4.1b)} \\ & Z_{AB}\mathbf{\Phi}\{1\} = \hat{I}, && \text{(4.1c)} \\ & \mathbf{\Phi}\{1\}x_0 \in \mathcal{P}, && \text{(4.1d)} \end{aligned}$$

where $\hat{I} := \begin{bmatrix} I & 0 \end{bmatrix}^\intercal$.

Remark 8. *We denote the number of rows of* $\mathbf{\Phi}$ *as* $N_\Phi := N_x(T+1) + N_uT$. *To shorten notation in what follows, we denote* $\mathbf{\Phi}_1 := \mathbf{\Phi}_{1:N_x,:}$, *and* $\mathbf{\Phi}_2 := (\mathbf{\Phi})_{N_x+1:,:}$.

For the remainder of the chapter we abuse notation and refer to the first block-column of the full matrix $\mathbf{\Phi}$–denoted as $\mathbf{\Phi}\{1\}$–as $\mathbf{\Phi}$, where we drop the term $\{1\}$ for convenience. From here on, we shall use *global MPC* to refer to (2.3), or equivalently, (4.1). We shall use *localized MPC* to refer to problem (4.1) with the additional locality constraints as per Definition 3, i.e., $\mathbf{\Phi} \in \mathcal{L}_d$. As discussed in previous chapters, localized MPC is less computationally complex than global MPC–also, for appropriately chosen locality constraints, it confers substantial scalability benefits. For this reason, here we explore conditions where we can keep the locality parameter d small without affecting the overall optimal performance of the system.

4.3 Global Performance for Localized MPC

In this section, we analyze the effect of locality constraints $\mathbf{\Phi} \in \mathcal{L}$ on the optimal performance of the MPC problem (2.7). We are especially interested in scenarios where localized MPC achieves *optimal global performance*, i.e., $f^* = f^*_{\mathcal{L}}$, where f^* and $f^*_{\mathcal{L}}$ are the solutions to the global MPC problem and localized MPC problem, respectively, for some state x_0.

First, we must analyze the space of available trajectories from state x_0 for both global and localized MPC. We denote an available trajectory $\mathbf{y} := \begin{bmatrix} \mathbf{x}_{1:T}^\intercal & \mathbf{u}^\intercal \end{bmatrix}^\intercal$.

Definition 6. *Trajectory set* $\mathcal{Y}(x_0)$ *denotes the set of available trajectories from state* x_0 *under dynamics* (2.1)*:*

$$\mathcal{Y}(x_0) := \{\mathbf{y} : \exists \mathbf{\Phi} \ s.t.\ Z_{AB}\mathbf{\Phi} = \hat{I},\ \mathbf{y} = \mathbf{\Phi}_2 x_0\}.$$

Localized trajectory set $\mathcal{Y}_{\mathcal{L}}(x_0)$ *denotes the set of available trajectories from state* x_0 *under dynamics* (2.1) *and locality constraint* $\mathbf{\Phi} \in \mathcal{L}_d$*:*

$$\mathcal{Y}_{\mathcal{L}}(x_0) := \{\mathbf{y} : \exists \mathbf{\Phi} \ s.t.\ Z_{AB}\mathbf{\Phi} = \hat{I},\ \mathbf{\Phi} \in \mathcal{L}_d,\ \mathbf{y} = \mathbf{\Phi}_2 x_0\}.$$

Lemma 7. *(Optimal global performance) Given an initial condition x_0, if the local communication constraint set $\mathcal{L}_d$ is chosen such that $\mathcal{Y}(x_0) = \mathcal{Y}_{\mathcal{L}}(x_0)$, then the localized MPC problem will attain optimal global performance.*

Proof. Global MPC problem (4.1) can be written as

$$\begin{aligned} \min_{\mathbf{y}} \quad & f(\mathbf{y}) \\ \text{s.t.} \quad & \mathbf{y} \in \mathcal{Y}(x(\tau)) \cap \mathcal{P}. \end{aligned}$$

Similarly, the localized MPC problem can also be written in this form by replacing $\mathcal{Y}(x(\tau))$ with $\mathcal{Y}_{\mathcal{L}}(x(\tau))$. Thus, if $\mathcal{Y}(x(\tau)) = \mathcal{Y}_{\mathcal{L}}(x(\tau))$, the two problems are equivalent and will have the same optimal values. □

Remark: This is a sufficient but not necessary condition for optimal global performance. Even if this condition is not satisfied, i.e., $\mathcal{Y}_{\mathcal{L}}(x_0) \subset \mathcal{Y}(x_0)$, the optimal global trajectory may be contained within $\mathcal{Y}_{\mathcal{L}}(x_0)$. However, this is dependent on objective f. Our analysis focuses on stricter conditions which guarantee optimal global performance for *any* objective function.

We now explore cases in which $\mathcal{Y}(x_0) = \mathcal{Y}_{\mathcal{L}}(x_0)$, i.e., the localized MPC problem attains optimal global performance. Localized trajectory set $\mathcal{Y}_{\mathcal{L}}(x_0)$ is shaped by the dynamics and locality constraints:

$$Z_{AB}\mathbf{\Phi} = \hat{I} \tag{4.2a}$$

$$\mathbf{\Phi} \in \mathcal{L}. \tag{4.2b}$$

To obtain a closed-form solution for $\mathcal{Y}_{\mathcal{L}}(x_0)$, we will parameterize these constraints. Two equivalent formulations are available. The dynamics-first formulation parameterizes constraint (4.2a), then (4.2b), and the locality-formulation parameterizes the constraints in the opposite order. The dynamics-first formulation clearly shows how local communication constraints affect the trajectory space; the locality-first formulation is less clear in this regard, but can be implemented in code with lower computational complexity than the dynamics-first formulation. We now derive each formulation.

Dynamics-first formulation

We first parameterize (4.2a), which gives a closed-form expression for trajectory set $\mathcal{Y}(x_0)$.

Lemma 8. *(Image space representation of trajectory set) The trajectory set from state x_0 is described by:*

$$\mathcal{Y}(x_0) = \{\mathbf{y} : \mathbf{y} = Z_p x_0 + Z_h X \lambda, \quad \lambda \in \mathbb{R}^{N_\Phi}\},$$

where $Z_p := (Z_{AB}^\dagger)_{N_x+1:,:}\hat{I}$ and $Z_h := (I - Z_{AB}^\dagger Z_{AB})_{N_x+1:,:}$; and the size of the trajectory set is

$$\dim(\mathcal{Y}(x_0)) = \operatorname{rank}(Z_h X).$$

If x_0 has at least one nonzero value, then $\dim(\mathcal{Y}(x_0)) = N_u T$.

Proof. We start by noticing that for any $\mathbf{\Phi}_x$ satisfying constraint (4.1d), $\mathbf{\Phi}_1 = I$. This is due to the structure of Z_{AB}. Also, constraint (4.1d) is always feasible, with solution space of dimension $N_u T$. To see this, notice that $\operatorname{rank}(Z_{AB}) = \operatorname{rank}\begin{bmatrix} Z_{AB} & \hat{I} \end{bmatrix}$ always holds, since Z_{AB} has full row rank due to the identity blocks on its diagonal; apply the Rouché-Capelli theorem [78] to get the desired result. Hence, we can parameterize the space of solutions $\mathbf{\Phi}$ as:

$$\mathbf{\Phi} = Z_{AB}^\dagger \hat{I} + (I - Z_{AB}^\dagger Z_{AB})\Lambda \tag{4.3}$$

where Λ is a free variable with the same dimensions as $\mathbf{\Phi}$. Since $\mathbf{\Phi}_1 = I$ always holds, we can omit the first N_x rows of (4.3). Hence,

$$\mathbf{\Phi}_2 = Z_p + Z_h \Lambda. \tag{4.4}$$

Combining (4.4) and the definition of $\mathbf{y}$, we have

$$\mathbf{y} = \mathbf{\Phi}_2 x_0 = Z_p x_0 + Z_h \Lambda x_0. \tag{4.5}$$

Making use of augmented state X, rewrite this as

$$\mathbf{y} = \mathbf{\Phi}_2 x_0 = Z_p x_0 + Z_h X \vec{\Lambda}. \tag{4.6}$$

This gives the desired expression for $\mathcal{Y}(x_0)$ and its size.

Notice that if x_0 has at least one nonzero value, then $\operatorname{rank}(Z_h X) = \operatorname{rank}(Z_h)$ due to the structure of X. All that is left is to show $\operatorname{rank}(Z_h) = N_u T$. First, note that $\operatorname{rank}(Z_{AB}) = N_x(T+1)$ due to the identity blocks on the diagonal of Z_{AB}. It follows that $\operatorname{rank}(Z_{AB}^\dagger) = \operatorname{rank}(Z_{AB}^\dagger Z_{AB}) = N_x(T+1)$. Thus, $\operatorname{rank}(I - Z_{AB}^\dagger Z_{AB}) = N_\Phi - N_x(T+1) = N_u T$. Recall that Z_h is simply $I - Z_{AB}^\dagger Z_{AB}$ with the first N_x rows removed; this does not result in decreased rank, since all these rows are zero (recall that $\mathbf{\Phi}_1$ is always equal to I). Thus, $\operatorname{rank}(Z_h) = \operatorname{rank}(I - Z_{AB}^\dagger Z_{AB}) = N_u T$. □

To write the closed form of localized trajectory set $\mathcal{Y}_{\mathcal{L}}(x_0)$, we require some definitions:

Definition 7. *Constrained vector indices $\mathfrak{L}$ denote the set of indices of $\mathbf{\Phi}_2$ that are constrained to be zero by the locality constraint* (4.2b)*, i.e.*

$$(\overrightarrow{\mathbf{\Phi}_2})_{\mathfrak{L}} = 0 \Leftrightarrow \mathbf{\Phi} \in \mathcal{L}.$$

Let $N_{\mathfrak{L}}$ be the cardinality of $\mathfrak{L}$.

We now parameterize (4.2b) and combine this with Lemma 8, which gives a closed-form expression for localized trajectory set $\mathcal{Y}_{\mathcal{L}}(x_0)$.

Lemma 9. *(Image space representation of localized trajectory set) Assume there exists some $\mathbf{\Phi}$ that satisfies constraints* (4.2a) *and* (4.2b)*. Then, the localized trajectory set from state x_0 is described by the following:*

$$\mathcal{Y}_{\mathcal{L}}(x_0) = \{\mathbf{y} : \quad \mathbf{y} = Z_p x_0 + Z_h X F^\dagger g + Z_h X(I - F^\dagger F)\mu, \quad \mu \in \mathbb{R}^{N_{\mathfrak{L}}}\},$$

where $F := (Z_h^{blk})_{\mathfrak{L},:}$ and $g := -(\overrightarrow{Z_p})_{\mathfrak{L}}$; and the size of the localized trajectory set is

$$\dim(\mathcal{Y}_{\mathcal{L}}(x_0)) = \operatorname{rank}(Z_h X(I - F^\dagger F)).$$

Proof. Using the augmented matrix of Z_h, we can write the vectorization of (4.4) as

$$\overrightarrow{\mathbf{\Phi}_2} = \overrightarrow{Z_p} + Z_h^{\text{blk}} \overrightarrow{\Lambda} \tag{4.7}$$

where $\overrightarrow{\Lambda}$ is a free variable. Incorporate locality constraint (4.2b) using the constrained vector indices:

$$(\overrightarrow{Z_p} + Z_h^{\text{blk}} \overrightarrow{\Lambda})_{\mathfrak{L}} = 0. \tag{4.8}$$

This is equivalent to $(\overrightarrow{Z_p})_{\mathfrak{L}} + (Z_h^{\text{blk}})_{\mathfrak{L},:} \overrightarrow{\Lambda} = 0$, or $F\overrightarrow{\Lambda} = g$. We can parameterize this constraint as

$$\overrightarrow{\Lambda} = F^\dagger g + (I - F^\dagger F)\mu \tag{4.9}$$

where μ is a free variable. Plugging this into (4.6) gives the desired expression for $\mathcal{Y}_{\mathcal{L}}(x_0)$ and its size. We remark that there is no need to consider $\mathbf{\Phi}_1 = I$ in relation to the locality constraints, since the diagonal sparsity pattern of the identity matrix (which corresponds to self-communication, i.e., node i 'communicating' to itself) satisfies any local communication constraint. □

Theorem 2. *(Optimal global performance) If x_0 has at least one nonzero value, then localized MPC attains optimal global performance if:*

1. *there exists some $\mathbf{\Phi}$ that satisfies constraints* (4.2a) *and* (4.2b)*, and*
2. $\operatorname{rank}(Z_h X(I - F^\dagger F)) = N_u T$.

Proof. By definition, $\mathcal{Y}_\mathcal{L}(x_0) \subseteq \mathcal{Y}(x_0)$. Equality is achieved if and only if the two sets are of equal size. Applying Lemmas 8 and 9 shows that the conditions of the theorem are necessary and sufficient for $\mathcal{Y}_\mathcal{L}(x_0)$ and $\mathcal{Y}(x_0)$ to be equal. Then, apply Proposition 7 for the desired result. □

This formulation provides intuition on how the inclusion of locality constraints affects the trajectory set–this is made clear in numerical examples in Section 4.3. However, checking the conditions of Theorem 2 is computationally expensive. In particular, we must assemble and compute the rank of matrix $Z_h X(I - F^\dagger F)$. The complexity of this operation is dependent on the size of the matrix, which increases as $\mathbf{\Phi}$ becomes more sparse (as enforced by locality constraints). This is a problem, since it is generally preferable to use very sparse $\mathbf{\Phi}$, as previously mentioned–this would correspond to an extremely large matrix $Z_h X(I - F^\dagger F)$, which is time-consuming to compute with. Ideally, sparser $\mathbf{\Phi}$ should instead correspond to *lower* complexity; this is the motivation for the next formulation.

Locality-first formulation

We first parameterize (4.2b), then (4.2a). This directly gives a closed-form expression for localized trajectory set $\mathcal{Y}_\mathcal{L}(x_0)$. We introduce some definitions that will prove helpful in the derivation of this formulation.

Definition 8. *Support vector indices $\mathfrak{M}$ denote the set of indices of $\overrightarrow{\mathbf{\Phi}}$ such that $(\overrightarrow{\mathbf{\Phi}})_\mathfrak{M} \neq 0$ is compatible with locality constraint* (4.2b)*. Let $N_\mathfrak{M}$ be the cardinality of $\mathfrak{M}$.*

Remark: this is complementary to Definition 7. Instead of looking at which indices are constrained to be zero, we now look at which indices are allowed to be nonzero. A subtlety is that this definition considers the entirety of $\mathbf{\Phi}$, while Definition 7 omits the first N_x rows of $\mathbf{\Phi}$.

Lemma 10. *(Image space representation of localized trajectory set) Assume there exists some* $\mathbf{\Phi}$ *that satisfies constraints* (4.2a) *and* (4.2b). *Then, the localized trajectory set from state* x_0 *is described by the following:*

$$\mathcal{Y}_{\mathcal{L}}(x_0) = \{\mathbf{y} : \quad \mathbf{y} = (X_2)_{:,\mathfrak{M}} H^\dagger k + (X_2)_{:,\mathfrak{M}} (I - H^\dagger H)\gamma, \quad \gamma \in \mathbb{R}^{N_{\mathfrak{M}}}\},$$

where $X_2 := (X)_{N_x+1:,:}$, $H = (Z_{AB}^{blk})_{:,\mathfrak{M}}$ *and* $k := \vec{I}$; *and the size of the localized trajectory set is*

$$\dim(\mathcal{Y}_{\mathcal{L}}(x_0)) = \operatorname{rank}((X_2)_{:,\mathfrak{M}}(I - H^\dagger H)).$$

Proof. Future trajectory $\mathbf{y}$ can be written as

$$\mathbf{y} = X_2 \overrightarrow{\mathbf{\Phi}} = (X_2)_{:,\mathfrak{M}} (\overrightarrow{\mathbf{\Phi}})_{\mathfrak{M}} \tag{4.10}$$

where the first equality arises from the definitions of $\mathbf{y}$ and X, and the second equality arises from the fact that zeros in $\overrightarrow{\mathbf{\Phi}}$ do not contribute to $\mathbf{y}$; thus, we only need to consider nonzero values $(\overrightarrow{\mathbf{\Phi}})_{\mathfrak{M}}$.

Using the augmented matrix of Z_{AB}, constraint (4.2a) can be rewritten as $Z_{AB}^{\text{blk}} \overrightarrow{\mathbf{\Phi}} = k$. Nonzero values $(\overrightarrow{\mathbf{\Phi}})_{\mathfrak{M}}$ must obey

$$H(\overrightarrow{\mathbf{\Phi}})_{\mathfrak{M}} = k. \tag{4.11}$$

Constraint (4.11) is feasible exactly when constraints (4.2a) and (4.2b) are feasible. By assumption, solutions exist, so we can parameterize the solution space as

$$(\overrightarrow{\mathbf{\Phi}})_{\mathfrak{M}} = H^\dagger k + (I - H^\dagger H)\gamma \tag{4.12}$$

where γ is a free variable. Substituting (4.12) into (4.10) gives the desired expression for $\mathcal{Y}_{\mathcal{L}}(x_0)$ and its size. □

Theorem 3. *(Optimal global performance) If* x_0 *has at least one nonzero value, then localized MPC attains optimal global performance if:*

1. *there exists some* $\mathbf{\Phi}$ *that satisfies constraints* (4.2a) *and* (4.2b), *and*

2. $\operatorname{rank}((X_2)_{:,\mathfrak{M}}(I - H^\dagger H)) = N_u T$.

Proof. Similar to Theorem 2; instead of applying Lemma 9, apply Lemma 10. □

To check the conditions of Theorem 3, we must assemble and compute the rank of matrix $(X_2)_{:,\mathfrak{M}}(I - H^\dagger H)$. The complexity of this operation is dependent on the size of this matrix, which decreases as $\mathbf{\Phi}$ becomes more sparse (as enforced by locality constraints). This is beneficial, since it is preferable to use very sparse $\mathbf{\Phi}$, which corresponds to a small matrix $(X_2)_{:,\mathfrak{M}}(I - H^\dagger H)$ that is easy to compute with. This is in contrast with the previous formulation, in which sparser $\mathbf{\Phi}$ corresponded to increased complexity. However, from a theoretical standpoint, this formulation does not provide any intuition on the relationship between the trajectory set $\mathcal{Y}(x_0)$ and the localized trajectory set $\mathcal{Y}_{\mathcal{L}}(x_0)$.

For completeness, we now use the locality-first formulation to provide a closed-form expression of $\mathcal{Y}(x_0)$. The resulting expression is equivalent to–though decidedly more convoluted than–the expression in Lemma 8:

Lemma 11. *(Image space representation of trajectory set) The trajectory set from state x_0 is described by:*

$$\mathcal{Y}(x_0) = \{\mathbf{y} : \quad \mathbf{y} = X_2 Z_{AB}^{blk} k + X_2(I - Z_{AB}^{blk\dagger} Z_{AB}^{blk})\gamma, \quad \gamma \in \mathbb{R}^{N_\Phi}\}.$$

Proof. In the absence of locality constraints, $\mathfrak{M}$ includes all indices of $\vec{\mathbf{\Phi}}$ since all entries are allowed to be nonzero. Here, $N_{\mathfrak{M}} = N_\Phi$, $(X_2)_{:,\mathfrak{M}} = X_2$, $(\vec{\mathbf{\Phi}})_{:,\mathfrak{M}} = \vec{\mathbf{\Phi}}$, and $H = Z_{AB}^{\text{blk}}$. Substitute these into the expression in Lemma 10 to obtain the desired result. □

Remark: By definition, $X_2 Z_{AB}^{\text{blk}} k = Z_p x_0$ and $X_2(I - Z_{AB}^{\text{blk}\dagger} Z_{AB}^{\text{blk}}) = Z_h X$. Substituting these quantities into Lemma 11 recovers Lemma 8.

Numerical example

To provide some intuition on the results from the previous subsections, we present a simple numerical example. We work with a system of three nodes in a chain interconnection, as shown in Fig. 4.1.

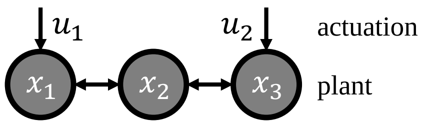

Figure 4.1: Example system with three nodes, two of which are actuated.

The system matrices are:

$$A = \begin{bmatrix} 1 & 2 & 0 \\ 3 & 4 & 5 \\ 0 & 6 & 7 \end{bmatrix}, \; B = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}. \tag{4.13}$$

We set initial state $x_0 = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^\top$, and choose a predictive horizon size of $T = 1$. Here, $N_\Phi = N_x(T+1) + N_uT = 8$. We choose locality constraints $\mathcal{L}$ such that each node may only communicate with its immediate neighbors; node 1 with node 2, node 2 with both nodes 1 and 3, and node 3 with node 2. Then, locality constraint (4.2b) is equivalent to

$$\mathbf{\Phi} = \begin{bmatrix} * & * & 0 \\ * & * & * \\ 0 & * & * \\ * & * & 0 \\ * & * & * \\ 0 & * & * \\ * & * & 0 \\ 0 & * & * \end{bmatrix} \tag{4.14}$$

where $*$ indicate values that are allowed to be nonzero. The support vector indices are $\mathfrak{M} = \{1, 2, 4, 5, 7, 9-16, 18, 19, 21, 22, 24\}$, and the constrained vector indices are $\mathfrak{L} = \{3, 5, 11, 14\}$ (recall that indices in $\mathfrak{L}$ do not include the first N_x rows of $\mathbf{\Phi}$). We confirm that there exists some $\mathbf{\Phi}$ that satisfies both dynamics constraints (4.2a) and locality constraints (4.2b) by checking that constraint (4.11) is feasible.

We start with dynamics-first formulation. In our case,

$$Z_h = \begin{bmatrix} 0_{5\times 3} & c_1 & 0_{5\times 1} & c_2 & c_1 & c_2 \end{bmatrix} \tag{4.15}$$

where c_1 and c_2 are defined as

$$c_1 := \frac{1}{2}\begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, c_2 := \frac{1}{2}\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \tag{4.16}$$

Z_h has a rank of 2. Then, $Z_hX = \begin{bmatrix} Z_h & Z_h & Z_h \end{bmatrix}$, also has a rank of 2. Per Lemma 8, this is the size of the trajectory set $\mathcal{Y}(x_0)$, which is exactly equal to N_uT, as expected.

$$Z_h X = \begin{bmatrix} 0_{5\times3} & c_1 & 0_{5\times1} & \boxed{c_2} & c_1 & \boxed{c_2} & 0_{5\times3} & c_1 & 0_{5\times1} & c_2 \\ c_1 & c_2 & 0_{5\times3} & \boxed{c_1} & 0_{5\times1} & c_2 & \boxed{c_1} & c_2 & & \end{bmatrix} \tag{4.17}$$

$$Z_h X(I - F^\dagger F) = \begin{bmatrix} 0_{5\times3} & c_1 & 0_{5\times2} & c_1 & 0_{5\times4} & c_1 & 0_{5\times1} & c_2 & c_1 & c_2 \\ 0_{5\times5} & c_2 & 0_{5\times1} & c_2 & & & & & & \end{bmatrix} \tag{4.18}$$

$$(X_2)_{:,\mathfrak{M}}(I - H^\dagger H) = \begin{bmatrix} 0_{5\times2} & c_1 & 0_{5\times1} & c_1 & 0_{5\times3} & c_1 & 0_{5\times1} & c_2 & c_1 & c_2 \\ 0_{5\times3} & c_2 & c_2 & & & & & & & \end{bmatrix} \tag{4.19}$$

We write out $Z_h X$ and $Z_h X(I - F^\dagger F)$ in full in equations (4.17) and (4.18). Boxed columns represent columns zeroed out as a result of locality constraints, i.e., if we replace the boxed columns in $Z_h X$ with zeros, we obtain $Z_h X(I - F^\dagger F)$. In our example, $Z_h X(I - F^\dagger F)$ also has a rank of 2; by Theorem 2, the local trajectory set is equal to the trajectory set, and by Theorem 2, the localized MPC problem attains optimal global performance.

Two observations are in order. First, we notice that the rank of $Z_h X$ (= 2) is low compared to the number of nonzero columns (= 12), especially when x_0 is dense. Additionally, the structure of $Z_h X$ is highly repetitive; the only two linearly independent columns are c_1 and c_2, and each appears 6 times in $Z_h X$. Furthermore, the specific values of x_0 do not affect the rank of these matrices–only the placement of nonzeros and zeros in x_0 matters. Second, notice that post-multiplying $Z_h X$ by $(I - F^\dagger F)$ effectively zeros out columns of $Z_h X$. However, due to the repetitive structure of $Z_h X$, this does not result in decreased rank for $Z_h X(I - F^\dagger F)$. In fact, it is difficult to find a feasible locality constraint that results in decreased rank. This observation is corroborated by simulations in Section 4.5, in which we find that locality constraints that are feasible also typically preserve global performance. For more complex systems and larger predictive horizons, post-multiplication of $Z_h X$ by $(I - F^\dagger F)$ no longer cleanly corresponds to zeroing out columns, but similar intuition applies.

We now apply the locality-first formulation. To check the conditions of Theorem 3, we must construct the matrix $(X_2)_{:,\mathfrak{M}}(I - H^\dagger H)$ and check its rank. This matrix is written out in equation (4.19). As expected, the rank of this matrix is also equal to two. Additionally, notice that $(X_2)_{:,\mathfrak{M}}(I - H^\dagger H)$ contains the same nonzero columns as $Z_h X(I - F^\dagger F)$: c_1 and c_2 are each repeated four times, in slightly different orders. This is unsurprising, as the two formulations are equivalent.

4.4 Algorithmic Implementation of Optimal Locality Selection

Leveraging the results of the previous section, we introduce an algorithm that selects the appropriate locality constraints for localized MPC. For simplicity, we restrict ourselves to locality constraints corresponding to d-local neighborhoods, though we remark that Subroutine 4 is applicable to arbitrary communication structures.

The localized MPC problem can be solved via distributed optimization techniques; the resulting distributed and localized MPC problem enjoys complexity that scales with locality parameter d, as opposed to network size N as illustrated in Chapter 2. Thus, when possible, it is preferable to use small values of d to minimize computational complexity. For a given system and predictive horizon length, Algorithm 4 will return the *optimal locality size* d–the smallest value of d that attains optimal global performance.

As previously described, the specific values of x_0 do not matter–only the placement of nonzeros and zeros in x_0 matters. We will restrict ourselves to considering dense values of x_0. For simplicity, our algorithm will work with the vector of ones as x_0–the resulting performance guarantees hold for *any* dense x_0.

To check if a given locality constraint preserves global performance, we must check the two conditions of Theorem 3. First, we must check whether there exists some $\mathbf{\Phi}$ that satisfies both dynamics and locality constraints; this is equivalent to checking whether (4.11) is feasible. We propose to check whether

$$\|H(H^\dagger k) - k\|_\infty \leq \epsilon \tag{4.20}$$

for some tolerance ϵ. Condition (4.20) can be distributedly computed due to the block-diagonal structure of H. Define partitions $[H]_i$ such that $H = \text{blkdiag}([H]_1, [H]_2 \dots [H]_N)$[2]. Then, (4.20) is equivalent to

$$\|[H]_i([H]_i^\dagger [k]_i) - [k]_i\|_\infty \leq \epsilon \quad \forall i. \tag{4.21}$$

If this condition is satisfied, then it remains to check the second condition of Theorem 3. To so, we must construct matrix $J := (X_2)_{:,\mathfrak{M}}(I - H^\dagger H)$ and check its rank. Notice that J can be partitioned into submatrices J_i, i.e., $J := \begin{bmatrix} J_1 & J_2 & \dots & J_N \end{bmatrix}$, where each block J_i can be constructed using only information from subsystem i, i.e., $[H]_i$, $[k]_i$, etc. Thus, J can be constructed in parallel–each subsystem i performs Subroutine 4 to construct J_i.

[2] H should have N_x blocks, where N_x is the number of states. Since $N \leq N_x$ and one subsystem may contain more than one state, we are able to partition H into N blocks as well.

Subroutine 4 Local sub-matrix for subsystem i

```
   inputs: $[H]_i$, $[k]_i$, $\epsilon$
   output: $J_i$
1: Compute $w = [H]_i^\dagger [k]_i$
2: if $\| [H]_i w - [k]_i \|_\infty > \epsilon$ :
     $J_i \leftarrow$ False
   else :
     $J_i \leftarrow I - [H]_i^\dagger [H]_i$
   return
```

Subroutine 4 checks whether the dynamics and locality constraints are feasible by checking (4.21), and if so, returns the appropriate submatrix J_i. Notice that the quantity $[H]_i$ is used in both the feasibility check and in J_i. Also, x_0 does not appear, as we are using the vector of ones in its place.

Having obtained J corresponding to a given locality constraint, we need to check its rank to verify whether global performance is preserved, i.e., $\text{rank}(J) = N_u T$, as per Theorem 3. As previously described, we restrict ourselves to locality constraints of the d-local neighborhood type, preferring smaller values of d as these correspond to lower complexity for the localized MPC algorithm 1. Thus, in Algorithm 4, we start with the smallest possible value of $d = 1$, i.e., subsystems communicate only with their immediate neighbors. If $d = 1$ does not preserve global performance, we iteratively increment d, construct J, and check its rank, until optimal global performance is attained.

Algorithm 4 Optimal local region size

```
   inputs: $A$, $B$, $T$, $\epsilon$
   output: $d$
   for $d = 1 \dots N$ :
1:   for $i = 1 \dots N$ :
       Construct $[H]_i$, $[k]_i$
       Run Subroutine 4 to obtain $J_i$
       if $J_i$ is False :
         continue
2:   Construct $J = \begin{bmatrix} J_1 & \dots & J_N \end{bmatrix}$
3:   if $\text{rank}(J) = N_u T$ :
       return $d$
```

In Step 1 of Algorithm 4, we call Subroutine 4 to check for feasibility and construct submatrices J_i. If infeasibility is encountered, or if optimal global performance is not attained, we increment d; otherwise, we return optimal locality size d.

Complexity

To analyze complexity, we first make some simplifying scaling assumptions. Assume that the number of states N_x and inputs N_u are proportional to the number of subsystems N, i.e., $O(N_x + N_u) = O(N)$. Also, assume that the number of nonzeros $N_{\mathfrak{M}}$ for locality constraint corresponding to parameter d are related to one another as $O(N_{\mathfrak{M}}) = O(NdT)$.

Steps 1 and 3 determine the complexity of the algorithm. In Step 1, each subsystem performs operations on matrix $[H]_i$, which has size of approximately $N_x(T+1)$ by dT–the complexity will vary depending on the underlying implementations of the pseudoinverse and matrix manipulations, but will generally scale according to $O((dT)^2 N_x T)$, or $O(T^3 d^2 N)$. In practice, d and T are typically much smaller than N, and this step is extremely fast; we show this in the next section.

In Step 3, we perform a rank computation on a matrix of size $(N_x + N_u)T$ by $N_{\mathfrak{M}}$. The complexity of this operation, if Gaussian elimination is used, is $O((N_x + N_u)^{1.38} T^{1.38} N_{\mathfrak{M}})$, or $O(T^{2.38} N^{2.38} d)$. Some speedups can be attained by using techniques from [79], which leverage sparsity–typically, J is quite sparse, with less than 5% of its entries being nonzero. In practice, Step 3 is the dominating step in terms of complexity.

We remark that this algorithm needs only to be run once offline for any given localized MPC problem. Given a system and predictive horizon, practitioners should first determine the optimal locality size d using Algorithm 4, then run Algorithm 1 to solve for the DLMPC problem.

4.5 Simulations

We first present simulations to supplement runtime characterizations of Algorithm 4 from the previous section. Then, we use the algorithm to investigate how optimal locality size varies depending on system size, actuation density, and prediction horizon length. We find that optimal locality size is primarily a function of actuation density. We also verify in simulation that localized MPC performs identically to global MPC when we use the optimal locality size provided by Algorithm 4, as expected. Code needed to replicate all simulations can be found at `https://github.com/flyingpeach/LocalizedMPCPerformance`. This code uses the SLS-MATLAB toolbox [59], which includes an implementation of Algorithm 4.[3]

[3]When implementing subroutine 4 in MATLAB, use of the backslash operator (i.e., `H\k`) is faster than the standard pseudoinverse function (i.e., `pinv(H)*k`). The backslash operator also produces

System and parameters

Simulations are performed on the same system as in Chapter 2; a two-dimensional square mesh, where each node represents a two-state subsystem that follows linearized and discretized swing dynamics

$$\begin{bmatrix}\theta(t+1)\\ \omega(t+1)\end{bmatrix}_i = \sum_{j\in\mathbf{in}_i(1)} [A]_{ij} \begin{bmatrix}\theta(t)\\ \omega(t)\end{bmatrix}_j + [B]_i[u]_i + [w]_i,$$

where $[\theta]_i$, $[\dot{\theta}]_i$, $[u]_i$ are the phase angle deviation, frequency deviation, and control action of the controllable load of bus i. The dynamic matrices are

$$[A]_{ii} = \begin{bmatrix}1 & \Delta t\\ -\frac{k_i}{m_i}\Delta t & 1-\frac{d_i}{m_i}\Delta t\end{bmatrix},\ [A]_{ij} = \begin{bmatrix}0 & 0\\ \frac{k_{ij}}{m_i}\Delta t & 0\end{bmatrix},\ \text{and } [B]_{ii} = \begin{bmatrix}0 & 1\end{bmatrix}^\intercal \text{ for all } i.$$

Connectivity among nodes is determined at random; each node connects to each of its neighbors with a 40% probability. The expected number of edges is $0.8 * n * (n-1)$. We consider only fully connected graphs. The parameters in bus i: m_i^{-1} (inertia inverse), d_i (damping) and k_{ij} (coupling term) are randomly generated and uniformly distributed between $[0,\ 2]$, $[0.5,\ 1]$, and $[1,\ 1.5]$, respectively. We set the discretization step $\Delta t = 0.2$, and define $k_i := \sum_{j\in\mathbf{in}_i(1)} k_{ij}$.

Under the given parameter ranges, the system is typically neutrally stable, with a spectral radius of 1. The baseline system parameters are $n = 5$ (corresponding to a 5×5 grid containing 25 nodes, or 50 states) and 100% actuation. Note: this does not correspond to ‘full actuation’ in the standard sense; it means that each subsystem (which contains 2 states) has one actuator–50% of states are actuated. We use a prediction horizon length of $T = 15$ unless otherwise stated.

Algorithm runtime

We plot the runtime of Algorithm 4 in Fig. 4.2 for different system sizes and horizon lengths. We separately consider runtimes for matrix construction (Step 1) and rank determination (Step 3). The former is parallelized, while the latter is not.

Runtime for matrix construction is extremely small. Even for the grid with 121 subsystems (242 states), this step takes less than a millisecond. Interestingly, matrix construction runtime also stays relatively constant with increasing network size, despite the worst-case runtime scaling linearly with N, as described in the previous section. This is likely due to the sparse structure of H. Conversely, matrix construction runtime increases with increasing horizon length.

J_i matrices that are as sparse as possible, which facilitates faster subsequent computations.

Runtime for rank determination dominates total algorithm runtime, and increases with both system size and horizon length. Rank determination runtime appears to increase more sharply with increasing horizon length than with increasing system size. Further runtime reductions may be achieved by taking advantage of techniques described in the previous section; however, even without additional speedups, the runtime is no more than 10 seconds for the grid with 242 states.

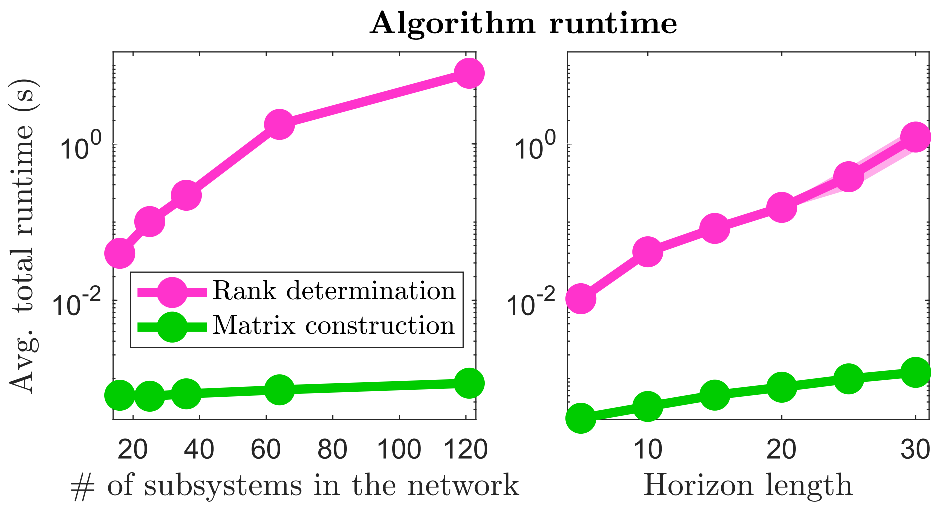

Figure 4.2: Runtime of matrix construction (Step 1, green) and rank determination (Step 3, pink) of Algorithm 4 vs. network size and horizon length. Parallelized (i.e., per-subsystem) runtimes are shown for matrix construction. The algorithm was run for grids containing 16, 25, 36, 64, and 121 subsystems. For each point, we run the algorithm on five different systems, and plot the average and standard deviation–here, the standard deviation is so small that it is barely visible. As expected, the rank determination step dominates total runtime, while the matrix construction step is extremely fast.

Optimal locality size as a function of system parameters

We characterize how optimal locality size changes as a function of the system size and horizon length–the results are summarized in Figure 4.3. Actuation density is the main factor that affects optimal locality size. Remarkably, at 100% actuation, the optimal locality size always appears to be $d = 1$, the smallest possible size (i.e., communication only occurs between nodes that share an edge). As we decrease actuation density, the required optimal locality size increases. This makes sense, as unactuated nodes must communicate to at least the nearest actuated node, and the distance to the nearest actuated node grows as actuation density decreases.

The optimal locality size also increases as a function of system size–but only when we do not have 100% actuation. At 60% actuation, for 121 nodes, the optimal locality size is around $d = 5$; this still corresponds to much less communication than global MPC.

Predictive horizon length does not substantially impact optimal locality size. At short horizon lengths ($T \leq 10$), we see some small correlation, but otherwise, optimal locality size stays constant with horizon size. Similarly, the stability of the system (i.e., spectral radius) appears to not affect optimal locality size; this was confirmed with simulations over systems with spectral radius of 0.5, 1.0, 1.5, 2.0, and 2.5 for 60%, 80%, and 100% actuation (not included in the plots).

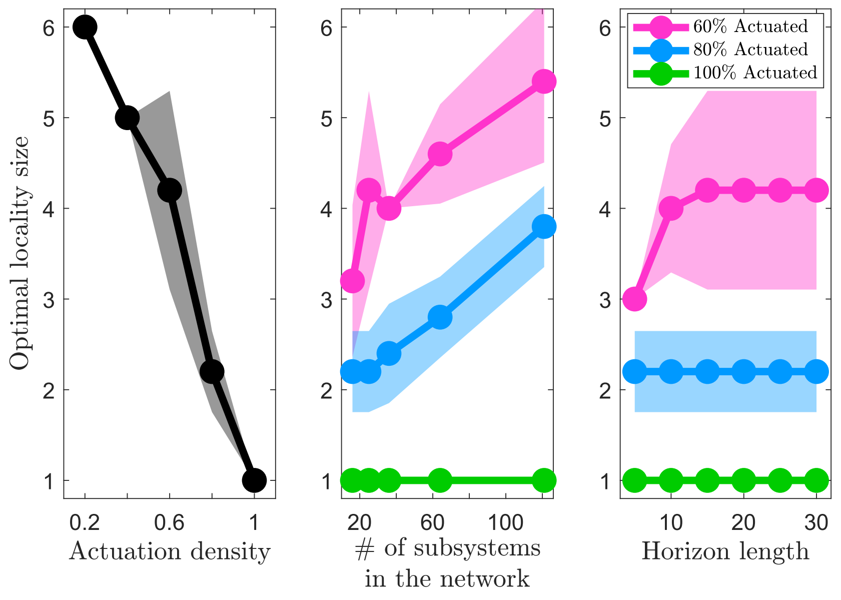

Figure 4.3: Optimal locality size as a function of various parameters. Each point represents the average over five different systems; standard deviations are shown by the fill area. ***(Left)*** Optimal locality size vs. actuation density. The two are inversely correlated. ***(Center)*** Optimal locality size vs. network size for 60% actuation (pink), 80% actuation (blue), and 100% actuation (green). For 60% and 80% actuation, optimal locality size roughly increases with network size. For 100% actuation, the optimal locality size is always 1, independent of network size. ***(Right)*** Optimal locality size vs. predictive horizon length for 60% actuation (pink), 80% actuation (blue), and 100% actuation (green). For 60% and 80% actuation, optimal locality size increases with horizon size up until $T = 10$, then stays constant afterward. For 100% actuation, the optimal locality size is always 1.

Optimal performance with locality constraints

From the previous section, we found that with 100% actuation, the optimal locality size is always 1. This means that even in systems with 121 subsystems, each node need only communicate with its immediate neighbors (i.e., 4 or less other nodes) to attain optimal global performance. This is somewhat surprising, as this is a drastic (roughly 30-fold) reduction in communication compared to global MPC. To confirm this result, we ran simulations on 20 different systems of size $N = 121$ and 100% actuation. We use LQR objectives with random positive diagonal matrices Q and R, and state bounds $\theta_i \in [-4, 4]$ for phase states and $\omega_i \in [-20, 20]$ for frequency states. We use random initial conditions where each value $(x_0)_i$ is drawn from a uniform distribution over $[-2, 2]$.

For each system, we run localized MPC with $d = 1$, then global MPC, and compare their costs over a simulation of 20 timesteps. Over 20 simulations, we find the maximum cost difference between localized and global MPC to be $5.6 \cdot 10^{-6}$. Thus, we confirm that the reported optimal locality size is accurate, since the cost of localized and global MPC are nearly identical.

Feasibility-optimality trade-off with locality constraints

In Algorithm 4, a given locality size is determined to be suboptimal if (1) the locality constraints are infeasible, or (2) the locality constraints are feasible, but matrix J has insufficient rank. The example from Section 4.3 suggests that the second case is rare–to further investigate, we performed 200 random simulations, in which all parameters were randomly selected from uniform distributions–grid size N from $[4, 11]$ (corresponding to system sizes of up to 121 subsystems), actuation density from $[0.2, 1.0]$, spectral radius from $[0.5, 2.5]$ and horizon length from $[3, 20]$. In these 200 simulations, we encountered 4 instances where a locality constraint was feasible but resulted in insufficient rank; in the vast majority of cases, if a locality constraint was feasible, the rank condition was satisfied as well.

4.6 Conclusion

In this work, we provided analysis and guarantees on locality constraints and global performance. We presented lemmas, theorems, and an algorithm to certify optimal global performance–these are the first results of their kind, to the best of our knowledge. We then leveraged these theoretical results to provide an algorithm that determines the optimal locality constraints that will expedite computation while preserving the performance–this is the first exact method to compute the optimal locality parameter d for DLMPC.

Several directions of future work may be explored:

1. The results in this work can be leveraged to investigate the relationship between network topology and optimal locality constraints, i.e., the strictest communication constraints that still preserve optimal global performance. Certain topologies may require long-distance communication between handful of nodes; others may require no long-distance communications. A more thorough characterization will help us understand the properties of systems that are suited for localized MPC.

2. Algorithm 4 considers d-local communication constraints; a natural extension is to consider non-uniform local communication constraints, which are supported by the theory presented in this work. A key challenge of this research direction is the combinatorial nature of available local communication configurations; insights from the research direction suggested above will likely help narrow down said set of configurations.

3. Simulations suggest that feasibility of a given locality constraint overwhelmingly coincides with optimal global performance. This poses the question of whether feasibility of a given locality constraint is *sufficient* for J to be full rank under certain conditions, and what these conditions may be. Additional investigation could reveal more efficient implementations of Algorithm 4, as bypassing the rank checking step would save a substantial amount of computation time.

4. This work focuses on nominal trajectories. Additional investigation is required to characterize the impact of locality constraints on trajectories robust to disturbances. For polytopic disturbances, the space (or minima) of available values of Ξg from problem (2.11) in Chapter 2 is of interest. Due to the additional variables in the robust MPC problem, we cannot directly reuse techniques from this chapter, though similar ideas may be applicable.

Chapter 5

DATA-DRIVEN APPROACH IN THE NOISELESS CASE

Abstract

In this chapter, we present a novel data-driven distributed control algorithm that is synthesized directly from trajectory data. Our method, data-driven Distributed and Localized Model Predictive Control (D^{3}LMPC), builds upon the data-driven System Level Synthesis (SLS) framework, which allows one to parameterize *closed-loop* system responses directly from collected open-loop trajectories. The resulting model-predictive controller can be implemented with distributed computation and only local information sharing. By imposing locality constraints on the system response, we show that the amount of data needed for our synthesis problem is independent of the size of the global system. Moreover, we show that our algorithm enjoys theoretical guarantees for recursive feasibility and asymptotic stability. Finally, we also demonstrate the optimality and scalability of our algorithm in a simulation experiment.

The content in this chapter has been published in [80].

5.1 Introduction

Contemporary large-scale distributed systems such as the Internet of Things enjoy ubiquitous sensing and communication, but are locally resource constrained in terms of power consumption, memory, and computation power. If such systems are to move from passive data-collecting networks to active distributed control systems, algorithmic approaches that exploit the aforementioned advantages subject to the underlying resource constraints of the network must be developed. Motivated by this emerging control paradigm, we seek to devise a distributed control scheme that is (a) model-free, eliminating the need for expensive system identification algorithms, and (b) scalable in implementation and computation. Our hypothesis is that for such systems, collecting local trajectory data from a small subset of neighboring systems is a far more feasible approach than deriving the intricate and detailed system models needed by model-based control algorithms. In this chapter, we show that such a data-driven distributed control approach can scalably provide optimal performance and constraint satisfaction, along with feasibility and stability guarantees.

Prior work

The majority of data-driven control approaches have focused on providing solutions to the linear quadratic regulator (LQR) problem. Among these works, we focus on the *direct methods* that do not require a system identification step [37]–[42]. Specifically, we highlight the work in [41], which applies behavioral systems theory to parametrize systems from past trajectories.[1] This idea has then given rise to several different data-enabled Model Predictive Control (MPC) approaches [43]–[46]. However, these approaches require gathering past trajectories of the global system, which hinders their scalability and challenges their applicability in the distributed setting. Even though some recent works have been developed where data-driven approaches were applied to the distributed setting, providing theoretical guarantees for these approaches remain difficult. For instance, the work in [47] provides an algorithm to solve the LQR problem where the dynamic matrices are unknown and communication only occurs at a local scale. This algorithm relies on the existence of "auxiliary" links among subsystems, which can make its extension to an online approach (eg. MPC) very costly. It is also unclear how theoretical guarantees can be derived for an MPC approach relying on these techniques. On the other hand, some approaches use data-driven formulations to provide theoretical guarantees (re-

[1]For a more in-depth treatment of behavioral system theory in the context of control problems, interested readers are referred to [81], [82], and the references therein.

cursive feasibility and asymptotic stability) in distributed MPC approaches where providing guarantees with conventional techniques is in general a hard problem and usually results in conservatism [10], [83]. However, in these cases knowledge of the system dynamics is assumed. It remains as an open question how to develop a *scalable* distributed MPC approach where the system model is unknown and only *local* measurements are available for each subsystem. It is important that such an approach enjoys the same theoretical guarantees of recursive feasibility and asymptotic stability as standard MPC approaches.

Contributions

We address this gap and present a data-driven version of the model-based Distributed Localized MPC (DLMPC) algorithm for linear time-invariant systems in a noise-free setting. We rely on recent results on data-driven SLS [46], which show that optimization problems over system-responses can be posed using only libraries of past system trajectories without explicitly identifying a system model. We extend these results to the localized and distributed setting, where subsystems can only collect and communicate information within a local neighborhood. In this way, the model-based DLMPC problem can equivalently be posed using only *local* libraries of past system trajectories, without explicitly identifying a system model. We then exploit this structure, together with the the separability properties of the objective function and constraints, and provide a distributed synthesis algorithm based on the Alternating Direction Method of the Multipliers (ADMM) [52] where only local information sharing is required. Hence, in the resulting implementation, each sub-controller solves a low-dimensional optimization problem defined over a local neighborhood, requiring only local data sharing and no system model. Since this problem is analogous to the model-based DLMPC problem in Chapter 2, our approach directly inherits its guarantees for convergence, recursive feasibility and asymptotic stability discussed in Chapter 3. Through numerical experiments, we validate these results and further confirm that the complexity of the subproblems solved at each subsystem does not scale relative to the full size of the system.

5.2 Problem Formulation

In Chapter 2, we provided an algorithmic solution to compute a *localized* MPC controller for the discrete-time linear time-invariant (LTI) system with dynamics (2.1) composed of N interconnected subsystems. Here, we extend these results to the case where the topology is $\mathcal{G}_{(A,B)}(\mathcal{V},\mathcal{E})$ is known, but the system model (A, B)

is unknown. In this setup, each subsystem i has access to a collection of past local state and input trajectories, and only local communication is possible among d-hop neighboring subsystems. We rely on the notion of locality in Definition 1, and we add to that the d-external set, which will be helpful in future derivations.

Definition 9. *For a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, the d-external set of subsystem i is defined as $\boldsymbol{ext}_i(d) := \left\{ v_j \mid \boldsymbol{dist}(v_j \rightarrow v_i) > d \in \mathbb{N} \right\}$, where $\boldsymbol{dist}(v_j \rightarrow v_i)$ denotes the distance between v_j and v_i, i.e., the number of edges in the shortest path connecting v_j to v_i.*

Similar to Chapter 4, we assume no noise is present in this case. Hence, our goal is to solve the MPC subroutine (4.1) subject to locality constraints, i.e., $\mathbf{\Phi} \in \mathcal{L}_d$ when the matrices A and B are unknown at the synthesis time. In the same spirit of Chapter 2, we work under mild structural assumptions such as Assumption 1.

In what follows we show problem (2.3) admits a distributed solution and implementation requiring only local data and no explicit estimate of the system model.

5.3 Data-driven System Level Synthesis

In this section, we introduce an abridged summary of the SLS extension to a data-driven formulation [46] based on the behavioral framework of Willems [84]. This section is adapted from §2 of [46]. In following sections, we build on these concepts to provide the necessary results to provide a tractable reformulation of problem (2.3).

Behavioral system theory [41], [81], [82], [84] offers a natural way of studying the behavior of a dynamical system in terms of its input/output signals. In particular, Willem's Fundamental Lemma [84] offers a parametrization of state and input trajectories based on past trajectories as long as the data matrix satisfies a notion of persistance of excitation.

Definition 10. *A finite-horizon signal* $\mathbf{x}$ *with horizon T is* persistently exciting *(PE) of order L if the Hankel matrix*

$$H_L(\mathbf{x}) := \begin{bmatrix} x(0) & x(1) & \dots & x(T-L) \\ x(1) & x(2) & \dots & x(T-L+1) \\ \vdots & \vdots & \ddots & \vdots \\ x(L-1) & x(L) & \dots & x(T-1) \end{bmatrix}$$

has full rank.

Lemma 12. *(Willem's Fundamental Lemma [84]) Consider the LTI system* (**??**) *with controllable* (A, B) *matrices, and assume that there is no driving noise. Let* $\{\tilde{\mathbf{x}}, \tilde{\mathbf{u}}\}$ *be the state and input signals generated by the system over a horizon* T*. If* $\tilde{\mathbf{u}}$ *is PE of order* $n + L$*, then the signals* $\mathbf{x}$ *and* $\mathbf{u}$ *are valid trajectories of length* L *of the system* (2.1) *if and only if*

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{u} \end{bmatrix} = H_L(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})\ g \ \textit{ for some } g \in \mathbb{R}^{T-L+1}, \tag{5.1}$$

where $H_L(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}) := \begin{bmatrix} H_L(\tilde{\mathbf{x}})^\intercal & H_L(\tilde{\mathbf{u}})^\intercal \end{bmatrix}^\intercal$.

A natural connection can be established between the data-driven parametrization (5.1) and the SLS parametrization (2.5). In particular, the achievability constraint (2.6) can be replaced by a data-driven representation by applying Willems' Fundamental Lemma [84] to the columns of the system responses. Given a system response, we denote the set of columns corresponding to subsystem i as $\mathbf{\Phi}^i$, i.e.,

$$\mathbf{\Phi} = \begin{bmatrix} \mathbf{\Phi}^1 & \mathbf{\Phi}^2 & \dots & \mathbf{\Phi}^N \end{bmatrix}.$$

The key insight is that, by definition of the system responses (2.5), $\mathbf{\Phi}^i_x$ and $\mathbf{\Phi}^i_u$ are the impulse response of $\mathbf{x}$ and $\mathbf{u}$ to $[x_0]_i$, which are themselves, valid system trajectories that can be characterized using Willemsâ Fundamental Lemma. This can be seen from the following decomposition of the trajectories

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{u} \end{bmatrix} = \mathbf{\Phi}[x_0]_i = \sum_{i=1}^{N} \mathbf{\Phi}^i [x_0]_i.$$

Lemma 13. *(Lemma 1 of [46]) Given the assumptions of Lemma 12, the set of feasible solutions to constraint* (2.6) *over a time horizon* $t = 0, 1, \dots, L-1$ *can be equivalently characterized as:*

$$H_L(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})\mathbf{G}, \ \textit{for all } \mathbf{G} \textit{ s.t. } H_1(\tilde{\mathbf{x}})\mathbf{G} = I. \tag{5.2}$$

Proof. The proof to this Lemma can be found in [46], and consists on showing the set equality:

$$\{\mathbf{\Phi} : \ Z_{AB}\mathbf{\Phi} = I\} = \{H_L(\mathbf{x}, \mathbf{u})\mathbf{G} : \mathbf{G} \text{ s.t. } H_1(\mathbf{x})\mathbf{G} = I\}.$$

□

The following Corollary follows naturally from Lemma 13, and will be useful later to provide locality results in this data-driven parametrization.

Corollary 3.1. *The following is true:*

$$\{\mathbf{\Phi}^i : Z_{AB}\mathbf{\Phi}^i = I^i\} = \{H_L(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})\mathbf{G}^i : \mathbf{G}^i \textit{ s.t. } H_1(\tilde{\mathbf{x}})\mathbf{G}^i = I^i\},$$

where I^i denotes the i-th block column of the identity matrix.

Proof. This follows directly from Lemma 13 by noting that the constraints can be separated column-wise. □

While this connection between SLS and the behavioral formulation does not offer an immediate benefit, we will build on it in the following sections to equip the data-driven parametrization (5.2) with locality constraints so as to provide a reformulation of the localized MPC subroutine (2.3).

5.4 Localized data-driven System Level Synthesis

In this section we present the necessary results that allow us to recast the constraints in (2.12) in a localized data-driven parametrization. We first provide a naive parametrization of system responses subject to locality constraints based on Lemma 13 in terms of $\mathbf{G}$. We then build on this parameterization and show that localized system responses can be characterized using only locally collected trajectories.

Locality constraints in data-driven System Level Synthesis

We start by rewriting the locality constraints using the data-driven parameterization (5.2).

Lemma 14. *Consider the LTI system* (2.1) *with controllable* (A, B) *matrices, where each subsystem i is subject to locality constraints* (2.4)*. Assume that there is no driving noise. Given the state and input trajectories* $\{\tilde{\mathbf{x}}, \tilde{\mathbf{u}}\}$ *generated by the system over a horizon T with* $\mathbf{u}$ *PE of order at least $n + L$, the following parametrization over* $\mathbf{G}$ *characterizes all possible d-localized system responses over a time span of $L - 1$:*

$$\begin{aligned} &H_L(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})\mathbf{G}, \textit{ for all } \mathbf{G} \textit{ s.t. } H_1(\tilde{\mathbf{x}})\mathbf{G} = I, \\ &\qquad H_L([\tilde{\mathbf{x}}]_i)\mathbf{G}^j = 0 \; \forall j \notin \boldsymbol{in}_i(d), \\ &\qquad H_L([\tilde{\mathbf{u}}]_i)\mathbf{G}^k = 0 \; \forall k \notin \boldsymbol{in}_i(d+1), \end{aligned} \tag{5.3}$$

for all $i = 1, \ldots, N$.

Proof. We aim to show that

$$\{\mathbf{\Phi} : Z_{AB}\mathbf{\Phi} = I, \mathbf{\Phi} \in \mathcal{L}_d\} = \{H_L(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})\mathbf{G} : \mathbf{G} \text{ s.t. } (5.3)\}.$$

($\subseteq$) First, suppose that $\mathbf{\Phi} \in \mathcal{L}_d$ satisfies that $Z_{AB}\mathbf{\Phi} = I$. From Lemma 13, we immediately have that there exists a matrix $\mathbf{G}$ s.t. $\mathbf{\Phi} = H_L(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})\mathbf{G}$. Thus, we need only verify that this $\mathbf{G}$ satisfies the linear constraint in (5.3). This follows directly from the assumption that $\mathbf{\Phi} \in \mathcal{L}_d$, which states that

$$H_L([\tilde{\mathbf{x}}]_i)\mathbf{G}^j = [\mathbf{\Phi}_x]_{ij} = 0 \; \forall j \notin \mathbf{in}_i(d),$$
$$H_L([\tilde{\mathbf{u}}]_i)\mathbf{G}^k = [\mathbf{\Phi}_u]_{ik} = 0 \; \forall k \notin \mathbf{in}_i(d+1).$$

Hence, $\mathbf{\Phi} \in$ RHS, proving this direction.

($\supseteq$) Now suppose that there exists a $\mathbf{G}$ that satisfies the constraints on the RHS and let $\mathbf{\Phi} = H_L(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})\mathbf{G}$. Since $H_1(\tilde{\mathbf{x}})\mathbf{G} = I$, from Lemma 2, we have that $\mathbf{\Phi}$ is achievable. From the other two constraints, we have that $\mathbf{\Phi} \in \mathcal{L}_d$, proving this direction and hence the lemma. □

It is important to note that even though Lemma 14 allows one to capture the locality constraint (2.4) by simply translating the locality constraints over $\mathbf{\Phi}$ to constraints over $\mathbf{G}$, it cannot be implemented with only local information exchange. In order to satisfy the constraints (5.3), each subsystem has to have access to global state and input trajectories and construct a global Hankel matrix. The PE condition of Lemma 12 further implies that the length of the trajectory that needs to be collected grows with the dimension of the global system state. In what follows we show how constraint (5.3) can further be relaxed to only require local information without introducing any additional conservatism.

Localized Data-driven System Level Synthesis

In this subsection we show that constraint (5.3) can be enforced (i) with local communication between neighbors, i.e., no constraints are imposed outside each subsystem d-neighborhood, and (ii) the amount of data needed, i.e., trajectory length, only scales with the size of the d-localized neighborhood, and not the global system. We start by providing a result that allows constraint (5.3) to be satisfied with local information only.

Definition 11. *Given a subsystem i satisfying the local dynamics*

$$[x(t+1)]_i = \sum_{j \in \{i, i\pm 1\}} [A]_{ij}[x(t)]_j + [B]_{ii}[u(t)]_i + [w(t)]_i, \tag{5.4}$$

we define its augmented d-localized subsystem *as the system composed by the states* $[x]_{\boldsymbol{in}_i(d+1)}$ *and augmented control actions* $[\bar{u}]_i := ([u]^{\mathsf{T}}_{\boldsymbol{in}_i(d+2)}\ [x]^{\mathsf{T}}_j)^{\mathsf{T}}$, $\forall j$ *s.t.* $\boldsymbol{dist}(j \to i) = d+2$*. That is, the system given by*

$$[x(t+1)]_{\boldsymbol{in}_i(d+1)} = [A]_{\boldsymbol{in}_i(d+1)}[x(t)]_{\boldsymbol{in}_i(d+1)} + [\bar{B}]_{\boldsymbol{in}_i(d+1)}[\bar{u}(t)]_i, \qquad (5.5)$$

with $\bar{B} := \begin{bmatrix}[B]_{\boldsymbol{in}_i(d+2)} & [A]_{ij}\end{bmatrix}\ \forall\ j$ *s.t.* $dist(j \to i) = d+2$.

Notice that by treating the state of the boundary subsystems as additional control inputs, we can view the augmented d-localized system as a standalone LTI system.

Lemma 15. *For* $i = 1, \dots, N$*, let* $\boldsymbol{\Psi}^i$ *be an achievable system response for the augmented d-localized subsystem* (5.5) *of subsystem i. Further assume that each* $\boldsymbol{\Psi}^i$ *satisfies constraints* (5.6)*:*

$$[\boldsymbol{\Psi}^i_x]_j = 0,\ \forall j \text{ s.t. } d+1 \le \boldsymbol{dist}(j \to i) \le d+2, \qquad (5.6a)$$

$$[\boldsymbol{\Psi}^i_u]_j = 0,\ \forall j \text{ s.t. } \boldsymbol{dist}(j \to i) = d+2 \qquad (5.6b)$$

for all i. Then, the system response $\boldsymbol{\Phi}$ *defined by* (5.7) *is achievable for system* (**??**) *and d-localized.*

$$[\boldsymbol{\Phi}]_{ij} := \begin{cases}[\boldsymbol{\Psi}^i]_j, & \forall j \in \boldsymbol{in}_i(d+1)\\ 0, & \textit{otherwise}\end{cases} \qquad (5.7)$$

for all $i = 1, \dots, N$ *is also achievable and d-localized.*

Proof. First, from the fact that $\boldsymbol{\Psi}^i$ is achievable for all $i = 1, \dots, N$, we have that $\Phi_x[0] = I$ by construction. Thus, to show that $\boldsymbol{\Phi}$ is achievable, it suffices to show that

$$\Phi_x[t+1] = A\Phi_x[t] + B\Phi_u[t], \quad \forall\, 0 \le t \le T-1.$$

We show this block-column-wise. Specifically, we show that the block columns Φ^i_x and Φ^i_u associated with each subsystem satisfy

$$\Phi^i_x[t+1] = A\Phi^i_x[t] + B\Phi^i_u[t], \quad \forall\, 0 \le t \le T-1. \qquad (5.8)$$

We further partition the rows of these block-columns into four subsets as follows:

$$\Phi^i_x = \begin{bmatrix}[\Phi^i_x]^{\mathsf{T}}_{\mathbf{in}_i(d)} & [\Phi^i_x]^{\mathsf{T}}_{d+1} & [\Phi^i_x]^{\mathsf{T}}_{d+2} & [\Phi^i_x]^{\mathsf{T}}_{\mathbf{ext}_i(d+2)}\end{bmatrix}^{\mathsf{T}},$$

where the notation $[\Phi^i_x]_k$ represents the entries of Φ_x corresponding to subsystems k-hops away from the i-th subsystem. Identical notation holds for the partition of Φ^i_u.

Using this partition, we have the following for Φ_x^i and $\Phi_u^i[t]$ given their definition in terms of $\mathbf{\Psi}$:

$$\Phi_x^i[t] = \begin{bmatrix} [\Psi_x^i[t]]_{\mathbf{in}_i(d)} \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \Phi_u^i[t] = \begin{bmatrix} [\Psi_u^i[t]]_{\mathbf{in}_i(d)} \\ [\Psi_u^i[t]]_{d+1} \\ 0 \\ 0 \end{bmatrix}.$$

We also partition the dynamics matrices A and B accordingly, where

$$A = \begin{bmatrix} A_{\mathbf{in}(d)}^{\mathbf{in}(d)} & A_{d+1}^{\mathbf{in}(d)} & 0 & 0 \\ A_{\mathbf{in}(d)}^{d+1} & A_{d+1}^{d+1} & A_{d+2}^{d+1} & 0 \\ 0 & A_{d+1}^{d+2} & A_{d+2}^{d+2} & A_{\mathbf{ext}_i(d+2)}^{d+2} \\ 0 & 0 & A_{d+2}^{\mathbf{ext}_i(d+2)} & A_{\mathbf{ext}_i(d+2)}^{\mathbf{ext}_i(d+2)} \end{bmatrix},$$

$$B = \begin{bmatrix} B_{\mathbf{in}(d)}^{\mathbf{in}(d)} & 0 & 0 & 0 \\ 0 & B_{d+1}^{d+1} & 0 & 0 \\ 0 & 0 & B_{d+2}^{d+2} & 0 \\ 0 & 0 & 0 & B_{\mathbf{ext}_i(d+2)}^{\mathbf{ext}_i(d+2)} \end{bmatrix}.$$

Here, the superscript represents an index on the block row and the subscript represents an index on the block column. The sparsity pattern of the partition follows directly from the definition of augmented d-localized subsystems and the subsystem dynamics (5.4).

We can now show that equation (5.8) holds for each of the ith block-columns and Φ^i is an achievable impulse response of the system. First, note that

$$\begin{aligned} [\Phi_x^i[t+1]]_{\mathbf{in}_i(d)} &= [A\Phi_x^i[t] + B\Phi_u^i[t]]_{\mathbf{in}_i(d)} \\ &= A_{\mathbf{in}_i(d)}^{\mathbf{in}_i(d)}[\Psi_x^i[t]]_{\mathbf{in}_i(d)} + B_{\mathbf{in}_i(d)}^{\mathbf{in}_i(d)}[\Psi_u^i[t]]_{\mathbf{in}_i(d)} + B_{d+1}^{\mathbf{in}_i(d)}[\Psi_u^i[t]]_{d+1} \\ &= [\Psi_x^i[t+1]]_{\mathbf{in}_i(d)}, \end{aligned}$$

where the second equality comes from the sparsity patterns of A, B, and $\mathbf{\Phi}^i$, and the third equality from the achievability of $\mathbf{\Psi}^i$. Similarly, to show that the boundary subsystems satisfy the dynamics, we note that

$$\begin{aligned} [\Phi_x^i[t+1]]_{d+1} &= A_{\mathbf{in}_i(d)}^{d+1}[\Psi_x^{(i)}(t)]_{\mathbf{in}_i(d)} + B_{d+1}^{d+1}[\Psi_u^{(i)}(t)]_{d+1} \\ &= [\Psi_x^{(i)}(t+1)]_{d+1} \\ &= 0. \end{aligned}$$

Lastly, from the sparsity pattern of the dynamic matrices and $\Phi^i[t]$, we trivially have that

$$[\Phi_x^{(i)}(t+1)]_{\mathbf{ext}_i(d)} = [A\Phi_x^i[t] + B\Phi_u^i[t]]_{\mathbf{ext}_i(d)} = 0,$$

concluding the proof for the achievability of $\mathbf{\Phi}$. We end by noting that $\mathbf{\Phi}$ is d-localized by construction. □

In light of this result, locality constraints as in Definition 2, i.e., $[\mathbf{\Phi}_x]_{ij} = 0 \;\forall\; i \notin \mathbf{out}_j(d)$, do not need to be imposed on every subsystem $i \notin \mathbf{out}_j(d)$. Instead, it suffices to impose this constraint only on subsystems i at a distance $d{+}2$ of subsystem j. Intuitively, this can be seen as a constraint on the propagation of a signal: if $[w]_j$ has no effect on subsystem i at distance $d+1$ because $[\mathbf{\Phi}_x]_{ij} = 0$, then the propagation of that signal is stopped and localized within that neighborhood. This idea will allow us to reformulate constraint (5.3) so that it can be imposed with only local communications.

However, despite the fact that locality constraints can now be achieved with local information exchanges, the amount of data that needs to be collected scales with the global size of the network n because we require that the control trajectory be at least PE of order at least $n+L$. In the following theorem, we build upon the previous results and show how this requirement can also be reduced to only depend on the size of a d-localized neighborhood.

Theorem 4. *Consider the LTI system* (2.1) *composed of subsystems* (5.4)*, each with controllable* $([A]_{\mathbf{in}_i(d+2)}, [B]_{\mathbf{in}_i(d+2)})$ *matrices for the augmented d-localized subsystem i. Assume that there is no driving noise and that the local control trajectory at the d-localized subsystem* $[\tilde{\mathbf{u}}]_{\mathbf{in}_i(d+1)}$ *is PE of order at least* $n_{\mathbf{in}_i(d)} + L$*, where* $n_{\mathbf{in}_i(d)}$ *is the dimension of* $[\tilde{\mathbf{x}}]_{\mathbf{in}_i(d)}$*. Then,* $\mathbf{\Phi}$ *is an achievable d-localized system response for each subsystem* (5.4) *if and only if it can be written as*

$$[\mathbf{\Phi}^i]_{\mathbf{in}_i(d)} = H_L([\tilde{\mathbf{x}}]_{\mathbf{in}_i(d+1)}, [\tilde{\mathbf{u}}]_{\mathbf{in}_i(d+1)})\mathbf{G}^i, \tag{5.9a}$$

$$[\mathbf{\Phi}^i]_{\mathbf{ext}_i(d+1)} = 0, \tag{5.9b}$$

where $\mathbf{G}^i$ *satisfies*

$$H_1([\tilde{\mathbf{x}}]_{\mathbf{in}_i(d+1)})\mathbf{G}^i = I^i, \tag{5.10a}$$

$$H_L([\tilde{\mathbf{x}}]_j)\mathbf{G}^i = 0 \;\forall i, j \text{ s.t. } d+1 \le \mathbf{dist}(j \rightarrow i) \le d+2, \tag{5.10b}$$

$$H_L([\tilde{\mathbf{u}}]_j)\mathbf{G}^i = 0 \;\forall i, j \text{ s.t. } \mathbf{dist}(j \rightarrow i) = d+2. \tag{5.10c}$$

Proof. ($\Rightarrow$) We first show that all d-localized system responses $\mathbf{\Phi}$ can be parameterized by a corresponding set of matrices $\{\mathbf{G}^i\}_{i=1}^N$. First, we note that since $\mathbf{\Phi}$ is d-localized, each d-localized subsystem impulse response $[\mathbf{\Phi}^i]_{\mathbf{in}_i(d+1)}$ is achievable on the augmented d-localized subsystem i. Thus, from applying Corollary 3.1, we have that there exists $\mathbf{G}^i$ satisfying constraint (5.10a) such that

$$[\mathbf{\Phi}^i]_{\mathbf{in}_i(d)} = H_L([\tilde{\mathbf{x}}]_{\mathbf{in}_i(d)}, [\tilde{\mathbf{u}}]_{\mathbf{in}_i(d+1)})\mathbf{G}^i.$$

Since $\mathbf{\Phi}$ is d-localized, we have that $\mathbf{G}^i$ satisfies both constraints (5.10b) and (5.10c), concluding the proof in this direction.
($\Leftarrow$) Now we show that if each $\mathbf{G}^i$ satisfies the constraint (5.10) for all $i = 1, \dots, N$, then the resulting $\mathbf{\Phi}$ is achievable and localized. Consider the augmented d-localized subsystem i and define

$$\mathbf{\Psi}^i = H_L([\tilde{\mathbf{x}}]_{\mathbf{in}_i(d)}, [\tilde{\mathbf{u}}]_{\mathbf{in}_i(d+1)})\mathbf{G}^i.$$

From Corollary 3.1, we have that $\mathbf{\Psi}^i$ is an achievable impulse response on the augmented d-localized subsystem i. Moreover, by construction it satisfies the sparsity condition in Lemma 15. Thus, constructing $\mathbf{\Phi}$ using $\mathbf{\Psi}$ as in equation (5.7) we conclude that $\mathbf{\Phi}$ is an achievable and d-localized system response. □

Corollary 4.1. *Consider a function $g : \mathbf{\Phi} \to \mathbb{R}$ such that*

$$g(\mathbf{\Phi}) = \sum_{i=1}^{N} g^i([\mathbf{\Phi}^i]_{\boldsymbol{in}_i(d+1)}).$$

Then, solving the optimization problem

$$\min\ g(\mathbf{\Phi})\ s.t.\ Z_{AB}\mathbf{\Phi} = I,\ \mathbf{\Phi} \in \mathcal{L}_d$$

is equivalent to solving

$$\min \sum_{i=1}^{N} g^i\Big(H_L([\tilde{\mathbf{x}}]_{\boldsymbol{in}_i(d)}, [\tilde{\mathbf{u}}]_{\boldsymbol{in}_i(d+1)})\mathbf{G}^i\Big)$$

$$s.t.\ \mathbf{G}^i \text{ satisfies } (5.10) \text{ for all } i = 1 \dots, N,$$

and then constructing the d-localized system response $\mathbf{\Phi}$ as per equation (5.9).

This result provides a data-driven approach in which locality constraints, as in equation (2.4), can be seamlessly considered and imposed by means of an affine subspace where only local information exchanges are necessary. Moreover, the amount of data needed to parametrize the behavior of the system does not scale with the size of the network but rather with d, the size of the localized region, which is usually much smaller than n. To the best of our knowledge, this is the first such result. As we show next, this will prove key in extending data-driven SLS to the distributed setting.

5.5 Distributed AND Localized algorithm for Data-driven MPC

In this section we make use of the results on localized data-driven SLS from previous sections and apply them to reformulate the MPC subproblem (2.3). We provide a distributed and localized algorithmic solution that does not scale with the size of the network. Lastly, we comment on the theoretical guarantees of this data-driven DLMPC (D^{3}LMPC) approach in terms of convergence, recursive feasibility and asymptotic stability.

System Level Synthesis reformulation of data-driven MPC

In light of Theorem 4, we can write the MPC subproblem (2.3) in terms of the variable $\mathbf{G}$ and localized Hankel matrices $H_L([\tilde{\mathbf{x}}]_{\mathbf{in}_i(d+2)}, [\tilde{\mathbf{u}}]_{\mathbf{in}_i(d+2)})$. To do this, we proceed as in reformulation (2.12) and rely on the equivalence between standard and data-driven SLS parametrizations, i.e., $\mathbf{\Phi} = H_L(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})\mathbf{G} \Leftrightarrow Z_{AB}\mathbf{\Phi} = I$. We make use of Lemma 4 to recast the locality constraints (2.4) into local affine constraints. Hence, we rewrite problem (2.3) as

$$\begin{aligned} \underset{\mathbf{\Phi}, \{\mathbf{G}^i\}_{i=1}^N}{\text{minimize}} \quad & f(\mathbf{\Phi} x_0) \\ \text{s.t.} \quad & x_0 = x(t),\ \mathbf{\Phi}_x x_0 \in \mathcal{X},\ \mathbf{\Phi}_u x_0 \in \mathcal{U}, \\ & [\mathbf{\Phi}^i]_{\mathbf{in}_i(d)} = H_L([\tilde{\mathbf{x}}]_{\mathbf{in}_i(d)}, [\tilde{\mathbf{u}}]_{\mathbf{in}_i(d+1)})\mathbf{G}^i, \\ & \mathbf{G}^i \text{ satisfies (5.10) } \forall i = 1, \dots, N. \end{aligned} \tag{5.11}$$

By introducing duplicate decision variables $\mathbf{\Phi}$ and $\mathbf{G}$, and rewriting the achievability and localization constraints in terms of the variable $\mathbf{G}$ by means of equation (5.10), the problem now enjoys a partially separable structure. In what follows, we make such structure explicit and take advantage of it to distribute the problem across different subsystems via ADMM.

A distributed subproblem solution via ADMM

To rewrite (5.11) and take advantage of the separability features in Assumption 1, we introduce the concept of row-wise separability and column-wise separability:

Definition 12. *Given the partition* $\{\mathfrak{r}_1, ..., \mathfrak{r}_k\}$*, a functional/set is* row-wise separable *if:*

- *for a functional,* $g(\mathbf{\Phi}) = \sum_{i=1}^{k} g_i\big(\mathbf{\Phi}(\mathfrak{r}_i, :)\big)$ *for some functionals* g_i *for* $i = 1, ..., k$.
- *for a set,* $\mathbf{\Phi} \in \mathcal{P}$ *if and only if* $\mathbf{\Phi}(\mathfrak{r}_i, :) \in \mathcal{P}_i$ $\forall i$ *for some sets* $\mathcal{P}_i$ *for* $i = 1, ..., k$.

An analogous definition exists for *column-wise separable* functionals and sets, where the partition $\{\mathfrak{c}_1, ..., \mathfrak{c}_k\}$ entails the columns of $\mathbf{\Phi}$, i.e., $\mathbf{\Phi}(:, \mathfrak{c}_i)$.

By Assumption 1 the objective function and the safety/saturation constraints in equation (5.11) are row-separable in terms of $\mathbf{\Phi}$. At the same time, the achievability and locality constraints (5.9), (5.10) are column-separable in terms of $\mathbf{G}$. Hence, the data-driven MPC subroutine becomes:

$$\begin{aligned}
\underset{\mathbf{\Phi}, \mathbf{\Psi}, \{\mathbf{G}^i\}_{i=1}^N}{\text{minimize}} \quad & \sum_{i=1}^{N} f^i([\mathbf{\Phi}]_i [x_0]_{\mathbf{in}_i(d)}) \\
\text{s.t.} \quad & x_0 = x(t),\ \mathbf{G}^i \text{ satisfies (5.10)}, \\
& [\mathbf{\Phi}_x]_i [x_0]_{\mathbf{in}_i(d)} \in \mathcal{X}^i,\ [\mathbf{\Phi}_u]_i [x_0]_{\mathbf{in}_i(d)} \in \mathcal{U}^i, \\
& [\mathbf{\Psi}^i]_{\mathbf{in}_i(d)} = H_L([\tilde{\mathbf{x}}]_{\mathbf{in}_i(d)}, [\tilde{\mathbf{u}}]_{\mathbf{in}_i(d+1)})\mathbf{G}^i \\
& \qquad\qquad\qquad\qquad \forall i = 1, \dots, N, \\
& \mathbf{\Phi} = \mathbf{\Psi}.
\end{aligned} \tag{5.12}$$

Notice that the objective function and the constraints decompose across the d-localized neighborhoods of each subsystem i. Given this structure, we can make use of ADMM to decompose this problem into row-wise local subproblems in terms of $[\mathbf{\Phi}]_i$ and column-wise local subproblems in terms of $\mathbf{G}^i$ and $\mathbf{\Psi}^i$, both of which can also be parallelized across the subsystems. The ADMM subroutine iteratively updates the variables as

$$[\mathbf{\Phi}]_i^{\{k+1\}} = \left\{ \begin{array}{ll} \underset{[\mathbf{\Phi}]_i}{\text{argmin}} & f^i([\mathbf{\Phi}]_i[x_0]_{\mathbf{in}_i(d)}) + \\ & \qquad \frac{\rho}{2}\left\| g^i(\mathbf{\Phi}, \mathbf{\Psi}^{\{k\}}, \mathbf{\Lambda}^{\{k\}}) \right\|_F^2 \\ s.t. & [\mathbf{\Phi}_x]_i[x_0]_{\mathbf{in}_i(d)} \in \mathcal{X}^i, \\ & [\mathbf{\Phi}_u]_i[x_0]_{\mathbf{in}_i(d)} \in \mathcal{U}^i, \quad x_0 = x(t). \end{array} \right\} \tag{5.13a}$$

$$[\mathbf{\Psi}^i]_{\mathbf{in}_i(d)}^{\{k+1\}} = \left\{ \begin{array}{l} \underset{[\mathbf{\Psi}^i]_{\mathbf{in}_i(d)}, \mathbf{G}^i}{\text{argmin}} \ \|g^{\mathbf{in}_i(d)}([\mathbf{\Phi}^i]^{\{k+1\}}, [\mathbf{\Psi}^i], [\mathbf{\Lambda}^i]^{\{k\}})\|_F^2 \\ \qquad s.t. \quad [\mathbf{\Psi}^i]_{\mathbf{in}_i(d)} = [H_L(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})]_{\mathbf{in}_i(d)}\mathbf{G}^i, \\ \qquad\qquad \mathbf{G}^i \text{ satisfies (5.10).} \end{array} \right\} \tag{5.13b}$$

$$[\mathbf{\Lambda}]_i^{\{k+1\}} = g^i(\mathbf{\Phi}^{\{k+1\}}, \mathbf{\Psi}^{\{k+1\}}, \mathbf{\Lambda}^{\{k\}}) \tag{5.13c}$$

where we define

$$g^*(\mathbf{\Phi}, \mathbf{\Psi}, \mathbf{\Lambda}) := [\mathbf{\Phi}]_* - [\mathbf{\Psi}]_* + [\mathbf{\Lambda}]_*$$

with $*$ denoting a subsystem or collection of subsystems, and

$$[H_L(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})]_{\mathbf{in}_i(d)} := H_L([\tilde{\mathbf{x}}]_{\mathbf{in}_i(d)}, [\tilde{\mathbf{u}}]_{\mathbf{in}_i(d+1)}).$$

We note that to solve this subroutine, and in particular optimization (5.13b), each subsystem only needs to collect trajectory of states and control actions of subsystems that are at most $d+2$ hops away. This subroutine thus constitutes a distributed and localized solution. We also emphasize that the trajectory length only needs to scale with the d-localized system size instead of the global system size. The full algorithm is given in Algorithm 5.

Theoretical guarantees

It is worth noting that this reformulation is equivalent to the closed-loop DLMPC also introduced in Chapter 2, with the achievability constraint $Z_{AB}\mathbf{\Phi} = I$ replaced by the data-driven parametrization in terms of $\mathbf{G}$ and the Hankel matrix H_L. For this reason, guarantees derived for the DLMPC formulation (3.1) directly apply to problem (5.11) when the constraint sets $\mathcal{X}$ and $\mathcal{U}$ are polytopes.

Convergence

Algorithm 5 relies on ADMM to separate both row, and column-wise computations, in terms of $\mathbf{\Phi}$ and $\mathbf{G}$, respectively. Each of these is then distributed into the subsystems in the network, and a communication protocol is established to ensure the ADMM steps are properly followed. Hence, one can guarantee the convergence

Algorithm 5 Subsystem's i implementation of D^{3}LMPC

```
    Input: tolerance parameters $\epsilon_p, \epsilon_d > 0$, Hankel matrices $[H_L(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})]_{\mathbf{in}_i(d+1)}$
    constructed from arbitrarily generated PE trajectories $\tilde{\mathbf{x}}_{\mathbf{in}_i(d+1)}, \tilde{\mathbf{u}}_{\mathbf{in}_i(d+1)}$.
1:  Measure local state $[x_0]_i$, $k \leftarrow 0$.
2:  Share measurement $[x_0]_i$ with $\mathbf{out}_i(d)$ and receive $[x_0]_j$ from $j \in \mathbf{in}_i(d)$.
3:  Solve optimization problem (5.13a).
4:  Share $[\mathbf{\Phi}]_i^{\{k+1\}}$ with $\mathbf{out}_i(d)$. Receive the corresponding $[\mathbf{\Phi}]_j^{\{k+1\}}$ from $\mathbf{in}_i(d)$
    and construct $[\mathbf{\Phi}^i]_{\mathbf{in}_i(d)}^{\{k+1\}}$.
5:  Perform update (5.13b).
6:  Share $[\mathbf{\Psi}^i]_{\mathbf{in}_i(d)}^{\{k+1\}}$ with $\mathbf{out}_i(d)$. Receive the corresponding $[\mathbf{\Psi}^j]_{\mathbf{in}_i(d)}^{\{k+1\}}$ from $j \in$
    $\mathbf{in}_i(d)$ and construct $[\mathbf{\Psi}]_i^{\{k+1\}}$.
7:  Perform update (5.13c).
8:  if $\left\|[\mathbf{\Psi}]_i^{\{k+1\}} - [\mathbf{\Phi}]_i^{\{k+1\}}\right\|_F \leq \epsilon_p$
    and $\left\|[[\mathbf{\Phi}]_i^{\{k+1\}} - [\mathbf{\Phi}]_i^{\{k\}}\right\|_F \leq \epsilon_d$:
      Apply computed control action:
      $[u_0]_i = \mathcal{H}_L([\tilde{\mathbf{u}}]_{\mathbf{in}_i(d)})\mathbf{G}^i[x_0]_{\mathbf{in}_i(d)}$, and return to step 1.
    else:
      Set $k \leftarrow +1$ and return to step 3.
```

of the data-driven version of DLMPC in the same way that convergence of model-based DLMPC is shown: by leveraging the convergence result of ADMM in [19]. For additional details see Lemma 6.

Recursive feasibility

Recursive feasibility for formulation (2.12) is guaranteed by means of a localized maximally invariant terminal set $\mathcal{X}_T$. This set can be computed in a distributed manner and with local information only as described in Chapter 3. In particular, a closed-loop map $\mathbf{\Phi}$ for the unconstrained localized closed-loop system has to be computed. In the model-based SLS problem with quadratic cost, a solution exists for the infinite-horizon case [71], which can be done in a distributed manner and with local information only. When no model is available, the same problem can be solved using the localized data-driven SLS approach introduced in §IV with a finite-horizon approximation, which also allows for a distributed synthesis with only local data. The length of the time horizon chosen to solve the localized data-driven SLS problem might slightly impact the conservativeness of the terminal set, but since the conservatism in the FIR approach decays exponentially with the horizon length, this harm in performance is not expected to be substantial for usual values of the horizon. Once $\mathbf{\Phi}$ for the unconstrained localized closed-loop system has been computed, Algorithm 2 can be used to synthesize this terminal set in a distributed and localized manner. Therefore, a terminal set that guarantees recursive feasibility for D^{3}LMPC can be computed in a distributed manner offline using only local information and without the need for a system model.

Asymptotic stability

Similar to recursive feasibility, asymptotic stability for the D^{3}LMPC problem (5.11) is directly inherited from the asymptotic stability guarantee for model-based DLMPC. In particular, adding a terminal cost based on the terminal set previously described is a sufficient condition to guarantee asymptotic stability of the DLMPC problem (3.1). Moreover, such cost can be incorporated in the D^{3}LMPC formulation in the same way as in the model-based DLMPC problem, and the structure of the resulting problem is analogous. This terminal cost introduces coupling among subsystems, but the coupling can be dealt with by solving step 3 of Algorithm 5 via local consensus. Notice that since step 3 of Algorithm 5 is written in terms of $\mathbf{\Phi}$, Algorithm 3 can be directly used to handle this coupling.

5.6 Simulation Experiments

We demonstrate through experiments that the D^{3}LMPC controller using only local data performs as well as a model-based DLMPC controller. We also show that for our algorithm, both runtime and the dimension of the data needed scale well with the size of the network.

Setup

We evaluate the performance and scalablity of our algorithm on a system composed of a chain of subsystems, i.e., that $\mathcal{E} = \{(i, i+1), (i+1, i),\ i = 1, ..., N-1\}$. Each subsystem i has a 2-dimensional state and takes a scalar control action u_i. We consider the same dynamics as in previous chapters where each of the subsystems follow linearized and discretized swing dynamics

$$\begin{bmatrix} \theta(t+1) \\ \omega(t+1) \end{bmatrix}_i = \sum_{j \in \mathbf{in}_i(1)} [A]_{ij} \begin{bmatrix} \theta(t) \\ \omega(t) \end{bmatrix}_j + [B]_i [u]_i + [w]_i,$$

where $[\theta]_i$, $[\dot{\theta}]_i$, $[u]_i$ are the phase angle deviation, frequency deviation, and control action of the controllable load of bus i. The dynamic matrices are

$$[A]_{ii} = \begin{bmatrix} 1 & \Delta t \\ -\frac{k_i}{m_i}\Delta t & 1 - \frac{d_i}{m_i}\Delta t \end{bmatrix}, \ [A]_{ij} = \begin{bmatrix} 0 & 0 \\ \frac{k_{ij}}{m_i}\Delta t & 0 \end{bmatrix}, \text{ and } [B]_{ii} = \begin{bmatrix} 0 & 1 \end{bmatrix}^\intercal \text{ for all } i.$$

The parameters m_i, d_i, k_{ij} are sampled uniformly at random from the intervals $[0, 2]$, $[0.5, 1]$, $[1, 1.5]$, respectively, and $k_i := \sum_{j \in \mathbf{in}_i(1)} k_{ij}$. Finally, the discretization time is set to be $\Delta t = 0.2$. The goal of the controller is to minimize the LQR cost with $Q = I$, $R = I$.

To show the optimality of our approach in Section 5.6, we consider a base system with 64 subsystems. To demonstrate the scalability of our method, we consider systems of varying sizes for our experiment in Section 5.6. Unless mentioned otherwise, for all the experiments, we consider a locality region of size $d = 2$, use a planning horizon of $T = 5$ steps and simulate the system forward for 30 steps.

Optimal performance

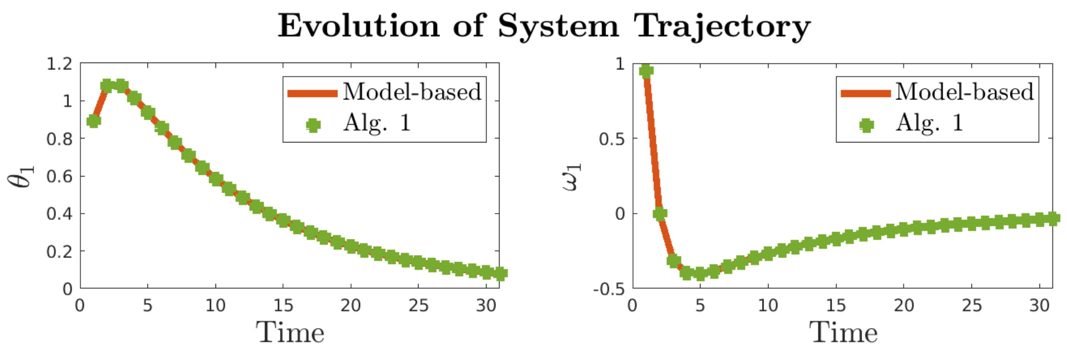

Figure 5.1: The trajectory generated by the model-based DLMPC algorithm (solid orange line) vs. the trajectory generated by D^{3}LMPC (green circles). We observe that the two coincides.

We evaluate the performance of D^{3}LMPC (Alg. 5) on the system described in Section 5.6. First, we show that the trajectory given by the D^{3}LMPC algorithm matches the trajectory generated by a model-based DLMPC solved with a centralized solver [60], [85]. Notice that the model-based DLMPC solves the optimization problem (2.12) with perfect knowledge of the system dynamics, while the D^{3}LMPC algorithm (Alg. 5) only has access to local past trajectories. Due to space constraints, we only show the state trajectory of the first subsystem (Figure 5.1). We observe that the trajectory generated by our controller matches the trajectory of the optimal controller with the same locality region size. Further, the optimal cost for both schemes is the same up to numerical precision. This confirms that the D^{3}LMPC algorithm (Alg. 5) can synthesize optimal controllers using only local trajectory data and no knowledge of the system dynamics.

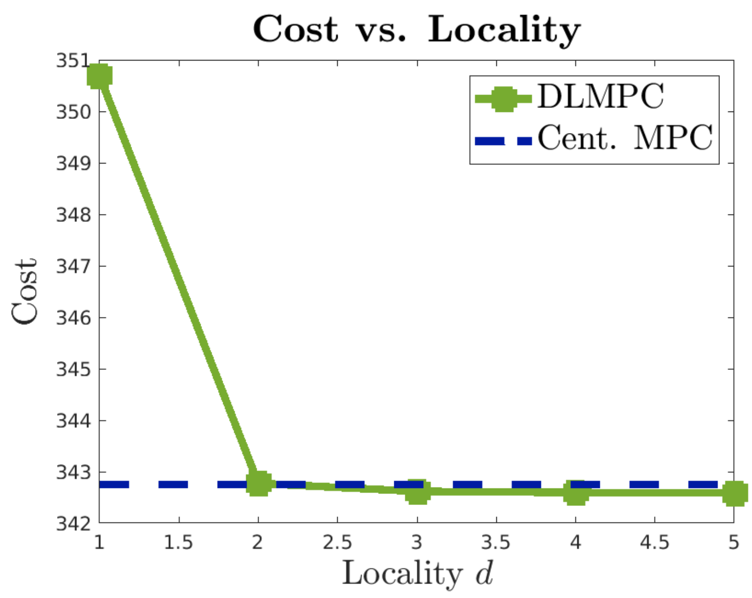

Figure 5.2: Optimal cost evolution with increase of the locality region. (Green) The cost achieved by the optimal controller versus the size of the locality region for the system response. (Blue) The cost achieved by a centralized MPC controller, i.e., that it has no locality constraints. We observe that cost for the distributed controllers decreases as the size of the locality region grows and approaches the cost of the centralized controller.

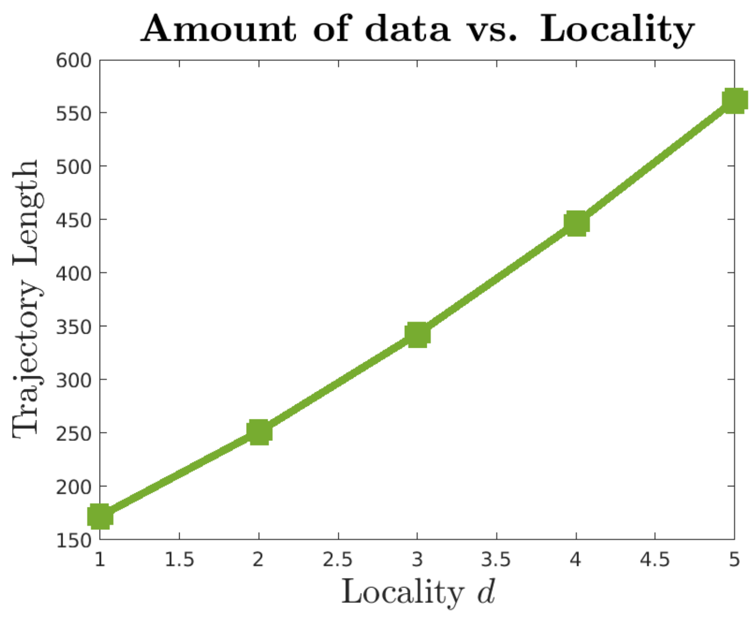

Figure 5.3: The growth of necessary length of collected trajectory versus the size of the locality region of system response. The trajectory length grows with the size of the locality region.

We further highlight the relevance of locality region size on the optimality of the solution. The size of the locality region d can be seen as a design parameter in Alg. 5 that allows one to tradeoff between computation complexity and performance of the controller. In Figure 5.2, we show how the optimal cost varies with the size of the locality region on the same system. As the size of the locality region grows, the optimal cost decreases. This matches the intuition that by allowing each subsystem to influence more subsystems, and as more information is made available to each subsystem, controllers of better quality can be synthesized. We note that the performance improvement by increasing the locality region size is the most significant when the locality region is small. In Figure 5.3 we simultaneously show how much the trajectory length needs to grow with the size of the locality region to satisfy the persistence of excitation condition for applying Willem's Fundamental Lemma. We note that the growth in the necessary trajectory length not only means longer trajectory needs to be collected, but also means that the size of the optimization problem grows, thus incurring higher computation complexity for each optimization step. Hence, the choice of an optimal d heavily depends on the specific application considered.

Scalability

First, we show that the runtime of our method scales well with the size of the global system. We consider systems composed of 9, 16, 36, 64, 81, 100, and 121 subsystems. For each system size, we randomly generate 10 different systems and report the average computation time per MPC step.[2] The result is shown in Figure 5.4. We note that the runtime only increased $2\times$ while the size of the system has increased more than $12\times$. Further, the growth of the runtime flattens as the size of the network grows, suggesting that our method scales well on sparsely connected systems. This trend has previously been observed with ADMM schemes for MPC [61].

[2]Runtime is measured after the first iteration to compute the runtime of the MPC algorithm after warmstart. The optimization problems were solved using the Gurobi [85] optimizer on a personal desktop computer with an 8-core Intel i7 processor.

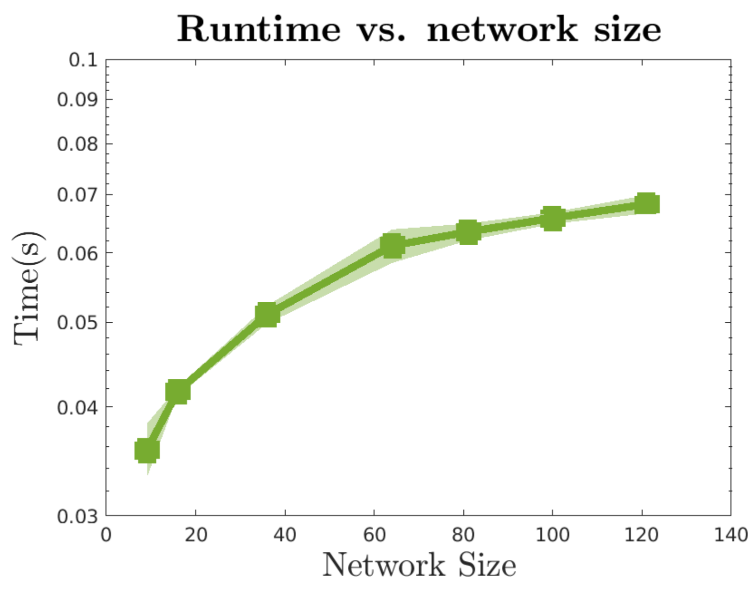

Figure 5.4: The average per-step per-subsystem runtime of the MPC algorithm. The solid line shows the average runtime over 10 randomly generated systems, and the circles represent the runtime for each of the 10 randomly generated instances for each system size.

Next, we show that the length of the trajectory that needs to be collected for the D^{3}LMPC controller grows much more slowly than that for a centralized data-driven method that does not exploit the locality structure of the problem (equivalent to solving the SLS problem with constraints (5.3) instead of (5.10)). The result is shown in Figure 5.5. We note that our method requires less data (length of the trajectory) to be collected in general. At the same time, the larger the system, the more benefit one gets from using our method over a centralized data-driven approach.

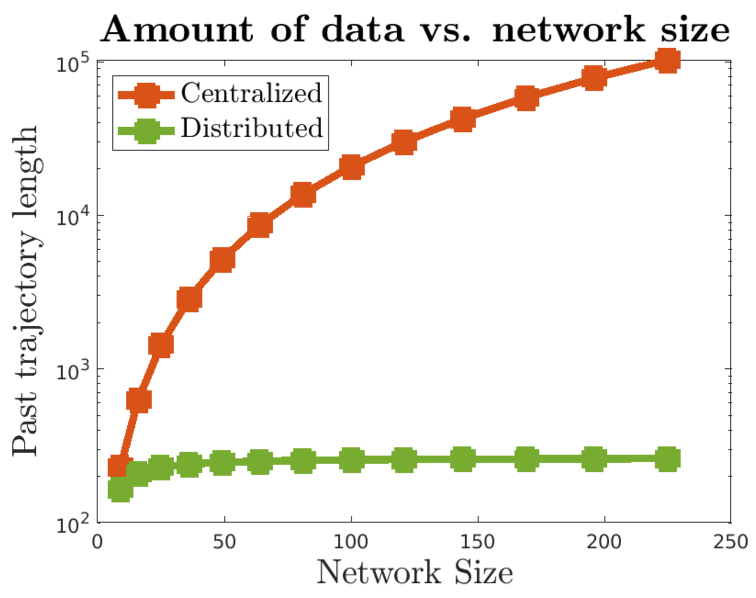

Figure 5.5: Length of necessary trajectory length versus network size. Note that this is plotted on a semilog axis. Our distributed approach requires much shorter trajectory over a centralized data-driven approach.

5.7 Conclusion

In this chapter we define and analyze a data-driven Distributed and Localized Model Predictive Control (D^{3}LMPC) scheme. This approach can synthesize optimal localized control policies using only local communication and requires no knowledge of the system model. We base our results on the data-driven SLS approach [46], and extend this framework to allow for locality constraints. We then use these results to provide an alternative data-driven synthesis for the DLMPC algorithm by exploiting the separability of the problem via ADMM. The resulting algorithm enjoys the same scalability properties as model-based DLMPC and only need trajectory data that scales with the size of the d-localized neighborhood. Moreover, recursive feasibility and stability guarantees that exist for model-based DLMPC directly apply to this framework.

The work presented here is, to the best of our knowledge, the first fully distributed and localized data-driven MPC approach that achieves globally optimal performance with local information collection and communication among subsystems. This, when extended to the noisy settings, offers a promising avenue forward towards localized and scalable learning and control with guarantees.

Chapter 6

EXPLICIT SOLUTION AND GPU PARALLELIZATION

Abstract

In this paper, we explore opportunities to accelerate the computation of MPC. Our contribution is two-fold. First, we provide an explicit solution for each of the subproblems resulting from the DLMPC scheme. We show that given the separability of the problem, the explicit solution is only divided into three regions per state and input instances, making the point location problem very efficient. Moreover, given the locality constraints, the subproblems are of much smaller dimension than the full problem, which significantly reduces the computational overhead of explicit solutions. Our method shows a large improvement in runtime per MPC iteration as compared with the results of computing the optimization with a solver online, and scales arbitrarily with the size of the network. Second, since the explicit solution does not require an optimization solver, we can parallelize the DLMPC routine in GPU. We exploit the locality constraints embedded in the DLMPC formulation to reduce the hardware-intrinsic communication overheads. Our parallel implementation achieves up to 50 times faster runtime than its CPU counterparts under various parameters. Furthermore, we find that the locality-aware GPU parallelization could halve the optimization runtime comparing to the naive acceleration. Overall, our results demonstrate the performance gains brought by software-hardware co-design with the information exchange structure in mind.

The content in this chapter has been published in [58] and [86].

6.1 Introduction

Model Predictive Control (MPC) has been shown to provide solutions for many industrial applications, but its applicability was long limited to slow processes, since solving an optimal control problem online imposes a large computational burden. In recent years, multiple solutions have been proposed to accelerate MPC runtimes [87]. One popular approach relies on providing computational enhancements to the optimization solving algorithms, either by exploiting the sparsity in the matrices or by finding initial points for the optimization [88]–[90]. In this realm, several works propose an explicit MPC strategy, where most of the computational burden is moved offline so the computational overhead of the online algorithms is greately reduced [91]. The other direction is to take advantage of state-of-the-art hardware such as multi-core processors (CPUs), many-core processors (GPUs) or field programmable arrays (FPGA) to perform computations in parallel [92]–[97]. In some instances these two approaches are combined, so efficient optimization algorithms are solved using multiple threads via hardware-specific implementations.

Prior work

On the algorithmic and computational side, explicit MPC has proven to be a very powerful technique. In explicit MPC approaches, the online computation reduces to providing the evaluation of a piecewise function by relying on the principles of multiparametric programming [48]–[50]. Despite its profound success, two important limitations restrict the applicability of MPC to large networks. On the one hand, explicit MPC has a limitation that is inherited from the computational complexity of multiparametric programming: finding a (piecewise) closed-form solution to an optimization problem becomes intractable for even modestly sized problems. On the other, even in the cases where the offline computation can be carried out, the solution is typically too complex to be evaluated efficiently online, in terms of both memory and runtime evaluation. These problems in applying MPC to large networks relate to the fact that, in the worst case, complexity increases exponentially with the number of constraints [98]. Efforts have been made to circumvent these issues. First, the complexity of the offline computation has been addressed by simplifying the MPC setup, using for example minimum-time formulations [99] or model reduction [100], among others. Secondly, efforts have also been made to tackle the online limitations, i.e., to facilitate efficient solutions to the point-location problem [101]–[103]. Once again, however, these methods are limited to systems of modest sizes and induce suboptimality in systems of large dimensions. Recent work has involved first formulating the network control problem as a distributed MPC problem, and then applying explicit MPC to the subproblems [4], [104]. Although these approaches work well for the intended application, they rely on heuristics and do not generalize well. Hence, the problem of how to make explicit MPC scalable, optimal, and applicable to large network settings remains an open question. This is especially relevant in distributed settings, where each subcontroller is typically repeatedly solving an optimization problem at every time step.

On the hardware side, although recently proposed methods provide promising avenues, most of their efforts are centered around achieving efficient computations by appropriately exploiting algorithmic features, and rely on hardware to simply parallelize mathematical operations. Hence, the hardware implementation of the algorithms is completely decoupled from the original system formulation, and therefore any hardware-intrinsic overhead can only be handled by using efficient programming practices. Yet, some branches of MPC directly encoding parallelization features in their formulation have received very little attention in this field. For instance, the merits of distributed MPC have been overlooked in parallel settings [87], despite the fact that distributed MPC formulations are very well-suited for parallelization. Moreover, our DLMPC framework allows to incorporate information exchange constraints among different subsystems. This feature is very relevant for hardware implementations since these information exchange constraints resemble the hardware-intrinsic communication limitations and overheads encountered in MPC parallelization. Despite the great promise of these MPC frameworks to deal with parallelization and hardware-intrinsic overheads in a principled manner through the problem formulation, its full potential has not been realized in the literature.

Contributions

In this chapter, we propose an explicit MPC solution that is applicable to large networks, and show how this enables a GPU pallelization of the DLMPC algorithm. We first provide an explicit solutions to the optimization problem for each subcontroller to solve. We show that the explicit solution requires just 3 partitions of the solution space per system state/input instance, thus making the point-location problem trivial when solving for each of the instances sequentially. Furthermore, we use this solution where algorithmic iterations result in basic arithmetic operations to provide a principled parallel implementation and overhead analysis. We exploit the potential for parallelization of the DLMPC scheme in a GPU, where the GPU is not used to parallelize arithmetic computations but rather each computing thread is tasked with computing the operations corresponding to a subsystem in the network. Moreover, we show that the limitations in communication among the GPU computing threads resemble the communication scheme in control systems for large-networks, and we take advantage of the local communication constraints that are already included in the DLMPC algorithm to explicitly deal with these hardware-intrinsic communication overheads in a principled manner. We demonstrate through simulations the effectiveness of our method.

6.2 Problem Formulation

Given the dynamics (2.1) with topology $\mathcal{G}_{(A,B)}$ as discussed in Chapter 2, consider the following MPC subroutine at time τ:

$$\begin{aligned} \min_{x_t,u_t,\gamma_t} \quad & \sum_{t=0}^{T} x_t^\mathsf{T} Q_t x_t + u_t^\mathsf{T} R_t u_t \\ \text{s.t.} \quad & x_0 = x(\tau),\ x_{t+1} = Ax_t + Bu_t, \\ & x_t^{min} \leq x_t \leq x_t^{max},\ u_t^{min} \leq u_t \leq u_t^{max}, \\ & [u_t]_i = \gamma_t^i([x_{0:t}]_j, [u_{0:t-1}]_j, [A]_{j,k}, [B]_{j,k}), \\ & t = 0, ..., T, \quad j,k \in \mathbf{in}_i(d), \quad i = 1, ..., N, \end{aligned} \tag{6.1}$$

where the matrices Q_t and R_t, and the upper and lower bounds x_t^{min}, u_t^{min}, x_t^{max}, u_t^{max} are chosen by design to not introduce coupling among subsystems for every t.

We recall Assumption 1, which adapted to the specific setting of problem (6.1) results in:

Assumption 2. *The cost function and constraints in formulation* (6.1) *are structured such that* $x_t^\mathsf{T} Q_t x_t = \sum_{i=1}^N [x_t]_i^\mathsf{T} [Q_t]_i [x_t]_i$*, and* $u_t^\mathsf{T} R_t u_t = \sum_{i=1}^N [u_t]_i^\mathsf{T} [R_t]_i [u_t]_i$*; and* $[x_t^{min}]_i \leq [x_t]_i \leq [x_t^{max}]_i$*, and* $[u_t^{min}]_i \leq [u_t]_i \leq [u_t^{max}]_i$ *for all* $i = 1, \ldots, N$*, and all* $t = 1, \ldots, T$.

Problem (6.1) can be solved via Algorithm 1 for feasible locality constraints as defined by the locality parameter d. However, the need for an optimization solver in step (3) degrades computational performance. Since all other steps in Algorithm 1 are solved in closed-form, an explicit analytical solution for step (3) is desirable to enhance computational performance and enable to solve problem (6.1) with simple operations only. Moreover, although Algorithm 1 is amenable to distribution, additional considerations are needed for a parallel implementation in real hardware. Particularly, memory handling in GPU is often an important bottleneck for performance. In this chapter, we explore these issues and provide an accelerated solution for problem (6.1), both algorithmically as well as for parallel hardware implementation.

6.3 An Explicit Solution for DLMPC

In this section, we derive an explicit solution for DLMPC. First, we use the results from Chapter 2 to reformulate problem (6.1) via the SLS parametrization. Next, we leverage the resulting optimization problem to provide an explicit analytic solution.

System Level Synthesis reformulation

By virtue of Chapter 2, the MPC subproblem (6.1) can equivalently be reformulated as:

$$\begin{aligned} \min_{\mathbf{\Phi}\{0\}} \quad & \|[C\ D]\mathbf{\Phi}\{0\}\,x_0\|_F^2 \\ \text{s.t.} \quad & x_0 = x(\tau),\ Z_{AB}\mathbf{\Phi}\{0\} = I, \\ & \begin{bmatrix} \mathbf{x}^{min} \\ \mathbf{u}^{min} \end{bmatrix} \leq \mathbf{\Phi}\{0\}\,x_0 \leq \begin{bmatrix} \mathbf{x}^{max} \\ \mathbf{u}^{max} \end{bmatrix}, \\ & \mathbf{\Phi}\{0\} \in \mathcal{L}_d, \end{aligned} \tag{6.2}$$

where the matrices C and D are constructed by arranging $Q_t^{\frac{1}{2}}$ and $R_t^{\frac{1}{2}}$ for all $t = 1, \dots, T$, respectively, in a block diagonal form. We are leveraging the result in [105], by which the cost function in problem (6.2) encodes for the H_2-norm of the system responses. For the remainder of the chapter, we will once again overload notation and write $\mathbf{\Phi}$ in place of $\mathbf{\Phi}\{0\}$, given that no driving noise is present, only the first block columns of the system responses need to be computed.

As shown in Chapter 2, problem (6.2) can be separated by virtue of Assumption 2, and distributed through ADMM [52]. The resulting distributed subroutine to be solved by each subcontroller i in the case of problem (6.1) becomes:

$$[\mathbf{\Phi}]_{i_r}^{k+1} = \left\{ \begin{aligned} \underset{[\mathbf{\Phi}]_{i_r}}{\text{argmin}} \quad & \left\|[\hat{C}]_i[\mathbf{\Phi}]_{i_r}[x_0]_{i_r}\right\|_F^2 + \frac{\rho}{2}\left\|[\mathbf{\Phi}]_{i_r} - [\mathbf{\Psi}]_{i_r}^k + [\mathbf{\Lambda}]_{i_r}^k\right\|_F^2 \\ \text{s.t.} \quad & \begin{bmatrix} \mathbf{x}^{min} \\ \mathbf{u}^{min} \end{bmatrix}_{i_r} \leq [\mathbf{\Phi}]_{i_r}[x_0]_{i_r} \leq \begin{bmatrix} \mathbf{x}^{max} \\ \mathbf{u}^{max} \end{bmatrix}_{i_r},\ [x_0]_{i_r} = [x(\tau)]_{i_r} \end{aligned} \right\} \tag{6.3a}$$

$$[\mathbf{\Psi}]_{i_c}^{k+1} = \left([\mathbf{\Phi}]_{i_c}^{k+1} + [\mathbf{\Lambda}]_{i_c}^k\right) + [Z_{AB}]_{i_c}^{\dagger}\Big([I]_{i_c} - [Z_{AB}]_{i_c}\left([\mathbf{\Phi}]_{i_c}^{k+1} + [\mathbf{\Lambda}]_{i_c}^k\right)\Big), \tag{6.3b}$$

$$[\mathbf{\Lambda}]_{i_r}^{k+1} = [\mathbf{\Lambda}]_{i_r}^k + [\mathbf{\Phi}]_{i_r}^{k+1} - [\mathbf{\Psi}]_{i_r}^{k+1}, \tag{6.3c}$$

where we define $[\hat{C}]_i := [[C]_i \ [D]_i]$, and the partitions $[\mathbf{\Phi}]_{i_r}$ and $[\mathbf{\Phi}]_{i_c}$ are chosen according to $\mathcal{L}_d$ for the rows and the columns of $\mathbf{\Phi}$, respectively.

Notice that subroutine (6.3) can be solved via Algorithm 1, where the subproblems solved by each subcontroller i are of dimension $d \ll N$. However, the step from subproblem (6.3) (step 3 in Algorithm 1) requires solving an optimization problem online, which is the bottleneck in terms of computational overhead. In the next subsection, we illustrate how we can provide an explicit analytical solution for subproblem (6.3).

Explicit solution

We start by introducing the following algebraic result:

Lemma 16. *Let Φ and a be row vectors, x_0 column vector of compatible dimension, and b_1, b_2 scalars. Then, the optimal solution to*

$$\begin{aligned} \underset{\Phi}{min} \quad & |\Phi x_0| + \frac{\rho}{2}\|\Phi - a\|_2^2 \\ s.t. \quad & b_2 \le \Phi x_0 \le b_1, \end{aligned} \tag{6.4}$$

is

$$\Phi^\star = \left(\rho a - \lambda x_0^\intercal\right) M, \tag{6.5}$$

where

$$\lambda = \begin{cases} \frac{\rho a M x_0 - b_1}{x_0^\intercal M x_0} & \text{if } \rho a M x_0 - b_1 > 0 \\ \frac{\rho a M x_0 - b_2}{x_0^\intercal M x_0} & \text{if } \rho a M x_0 - b_2 < 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad M := \left(2x_0 x_0^\intercal + \rho I\right)^{-1}.$$

Proof. Apply the KKT conditions to optimization (6.4). In particular, the stationarity condition is:

$$\nabla_\Phi\Big(\left|\Phi^\star x_0\right| + \frac{\rho}{2}\left\|\Phi^\star - a\right\|_2^2\Big) + \lambda_1 \nabla_\Phi\Big(\Phi^\star x_0 - b_1\Big) + \lambda_2 \nabla_\Phi\Big(-\Phi^\star x_0 + b_2\Big) = 0,$$

where λ_1 and λ_2 represent two scalar Lagrange multipliers whose values are unknown. This leads to the following result for the optimal Φ as a function of the unknown λ_1 and λ_2:

$$\Phi^\star = \left(\rho a - (\lambda_1 - \lambda_2) x_0^\intercal\right)\left(2x_0 x_0^\intercal + \rho I\right)^{-1}. \tag{6.6}$$

Notice that by Slater's condition (Chapter 5 in [68]) strong duality holds for problem (6.4) unless $b_1 = b_2$, in which case a closed form can be found directly with a proximity operator (we omit this degenerated case in the following discussion). Hence, we can make use of the dual problem to find the optimal solution. The dual problem can be written as:

$$\max_{\lambda_1,\lambda_2\geq 0} \left|\Phi^\star x_0\right| + \frac{\rho}{2}\left\|\Phi^\star - a\right\|_2^2 - \lambda_1(b_1 - \Phi^\star x_0) - \lambda_2(-b_2 + \Phi^\star x_0).$$

After substituting $\Phi^\star$ into the dual problem above, the cost function becomes a quadratic function of $\lambda := [\lambda_1\ \lambda_2]^\top$. In particular, after some algebraic manipulations one can conclude that the dual problem is a quadratic program equivalent to:

$$\max_{\lambda\geq 0}\ \lambda^\top c_2\lambda + c_1\lambda, \tag{6.7}$$

where $c_2 = \frac{1}{2}x_0^\top Mx_0\begin{bmatrix}-1 & 1\\ 1 & -1\end{bmatrix}$ and $c_1 = [\rho aMx_0 - b_1 \quad -\rho aMx_0 + b_2]$.

In order to compute the value of λ, we exploit complementary slackness:

$$\lambda_1(\Phi x_0 - b_1) = 0, \quad \text{and} \quad \lambda_2(-\Phi x_0 + b_2) = 0.$$

This condition makes evident that λ_1 and λ_2 cannot be both nonzero, since by assumption $b_1 < b_2$. Hence, let us assume without loss of generality that $\lambda_2 = 0$. The solution to problem (6.7) for λ_1 is as follows:

$$\lambda_1 = \begin{cases} \frac{\rho aMx_0 - b_1}{x_0^\top Mx_0} & \text{if } \lambda_1 > 0,\\ 0 & \text{otherwise.}\end{cases}$$

The form for $\lambda_1 = 0$ follows a similar structure. Notice that the matrix M is by definition positive definite. Hence, $x_0^\top Mx_0 > 0$ for all $x_0 \neq 0$ and the sign of λ_1 is purely determined by the sign of $aMx_0 - b_1$. This allows us to define the closed form solution for λ, and therefore for Φ, in a piecewise manner depending on the region. The criteria are specified in Table 6.1.

Recall that from optimization (6.4), the problem is only feasible if $b_1 < b_2$, hence the regions defined in Table 6.1 are disjoint and well-defined. Leveraging the entries of Table 6.1 and equation (6.6), one can find the explicit solution (6.5). □

Remark 9. *Given the structure of the matrix* $2x_0x_0^\top + \rho I$*, M can be computed in a very efficient manner using the Sherman–Morrison formula.*

We now apply Lemma 16 to subproblem (6.3a). Given the separability properties of the Frobenius norm and the constraints, this optimization problem can further be

Region in which x_0 lies	Corresponding solution for λ
$\rho a M x_0 - b_1 > 0$	$\lambda_1 = \frac{\rho a M x_0 - b_1}{x_0^\intercal M x_0}, \lambda_2 = 0$
$-\rho a M x_0 + b_2 > 0$	$\lambda_1 = 0, \lambda_2 = \frac{-\rho a M x_0 + b_2}{x_0^\intercal M x_0}$
$\rho a M x_0 - b_1 < 0$, $-\rho a M x_0 + b_2 < 0$	$\lambda_1 = 0, \lambda_2 = 0$

Table 6.1: Partition of the space of x_0 into the different regions that lead to different solutions for λ.

separated into *single rows* of $[\mathbf{\Phi}]_{i_r}$ and $[\mathbf{\Psi}]_{i_r}^k - [\mathbf{\Lambda}]_{i_r}^k$. Notice that this is true for the first term of the objective function as well, since $[\hat{C}]_i$ is a diagonal matrix by Assumption 2, so its components can be treated as factors multiplying each of the rows accordingly.

It is important to note that by definition of $[\mathbf{\Phi}]_{i_r}$, $[\mathbf{x}]_i = [\mathbf{\Phi}]_{i_r}[x_0]_{i_r}$. Hence, each row of $[\mathbf{\Phi}]_{i_r}$ multiplied with $[x_0]_{r_i}$ precisely corresponds to a given component of the a state or input instance, i.e., $[x_t]_i$ or $[u_t]_i$. We can now consider one of the single-row subproblems resulting from this separation and rename its variables, where Φ represents the given row of $[\mathbf{\Phi}]_{i_r}$, a represents the corresponding row of $[\mathbf{\Psi}]_{i_r}^k - [\mathbf{\Lambda}]_{i_r}^k$ and x_0, b_1 and b_2 correspond to the elements of $[x_0]_{\mathfrak{r}_i}$, $[x_t^{min}]_i/[u_t^{min}]_i$ and $[x_t^{max}]_i/[u_t^{max}]_i$, respectively. Without loss of generality we set each nonzero component of $[\hat{C}]_i$ to be equal to 1. By noting that for the inner product Φx_0, it holds that $\|\Phi x_0\|_F^2 = |\Phi x_0|$, and for any vector, the Frobenius norm is equivalent to the 2-norm, i.e., $\|\Phi - a\|_F^2 = \|\Phi - a\|_2^2$, we can directly apply Lemma 16 to each of the single-row subproblems in which the problem (6.3) can be separated. Hence, by Lemma 16, an explicit solution exists for optimization (6.3a). Thus, step 3 in Algorithm 1 can be solved explicitly for problem (6.1). Notice that all other computation steps in Algorithm 1 have closed-form solutions.

Computational complexity: In terms of complexity reduction, the solution presented in 16 consists of a point location problem followed by a matrix multiplication. The complexity of solving for each row of $[\mathbf{\Phi}]_{i_r}$ results in $O(d^2)$, since the point location problem involves only 3 regions and the size of the matrices is $O(d^2)$. Given that each subsystem performs this operation sequentially for each of the rows in $[\mathbf{\Phi}]_{i_r}$, the complexity of subproblem (6.3a) is also $O(d^2T)$. This is in contrast with the general solution in Chapter 2, where step 3 consists of solving an optimal problem with $O(d^2T)$ optimization variables and $O(dT)$ constraints. The significant overhead reduction given by the explicit solution (6.5) is due to the simplicity of the point location problem since the space of the solution is partitioned into 3 regions per state/input instances independently of the size of the global system N, the size of the locality region d and the total number of constraints. Hence, the complexity is only dominated by the matrix multiplication needed to compute the explicit solution $\Phi^{\star}$.

Differences with standard Explicit MPC: Contrary to **Conv**entional explicit MPC (where regions are computed offline and the online problem reduces to a point location problem), our approach is to solve step 3 in Algorithm 1 explicitly. This ensures that all the steps in the algorithm are solved in closed form or via an explicit solution, hence we refer to our approach as explicit MPC. Another difference between standard explicit MPC and our formulation is that, in our case, the regions are not defined by polytopes of x_0 [48]. In our case, M depends on x_0, thus, at each MPC iteration, new regions are computed as an explicit function of x_0, and for the subsequent ADMM iterations (within each MPC iteration) the parameter of the optimization problem is the corresponding row of $[\mathbf{\Psi}]_{i_r}^k - [\mathbf{\Lambda}]_{i_r}^k$ denoted as a in (6.5). The regions are indeed affine with respect to this parameter. Note that x_0 remains fixed within each MPC iteration. This idea is illustrated in Figure 6.1, where we illustrate the different regions involved in the computation of a given row of $[\mathbf{\Phi}]_{i_r}$ for two MPC iterations. In order to not overload notation, in this example we denote a single row of matrices $[\mathbf{\Phi}]_{i_r}$ and $[\mathbf{\Psi}]_{i_r}^k - [\mathbf{\Lambda}]_{i_r}^k$ with the same notation as the whole matrices themselves, i.e., $[\mathbf{\Phi}]_{i_r}$ and $[\mathbf{\Psi}]_{i_r}^k - [\mathbf{\Lambda}]_{i_r}^k$ denote a row of the homonymous matrices.

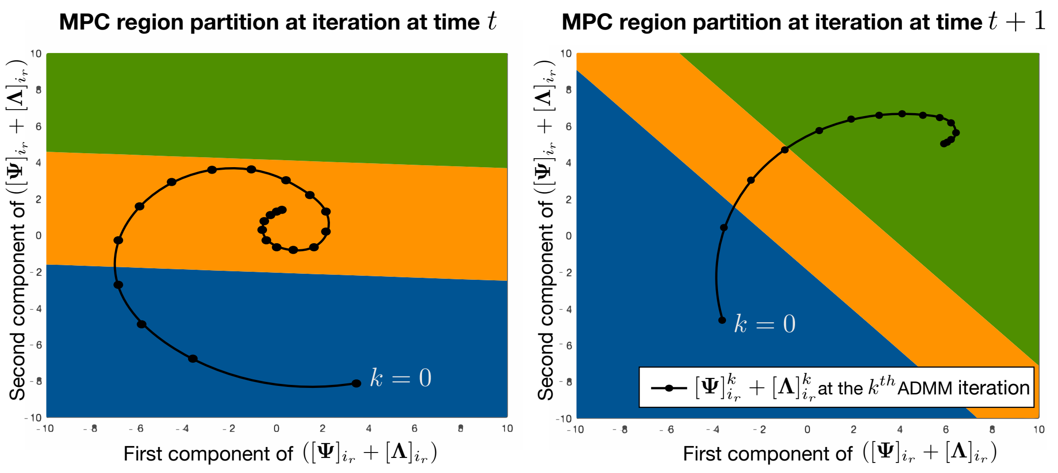

Figure 6.1: Illustration of the regions and parameter location over two MPC iterations, and the necessary ADMM iterations until **Conv**ergence in each of the MPC iterations. For simplicity in the representation, we consider the parameters in two dimensions.

6.4 A GPU Parallelization for DLMPC

In this section, we provide a GPU implementation for the explicit solution to DLMPC. We first discuss several important considerations of GPU hardware, and then provide strategy to incrementally enhance the implementation of DLMPC. We finish by discussing how locality constraints offer a unique advantage in GPU implementation.

GPU parallelization overview

GPU differs from CPU in computation and memory, which have a profound impact on programming and implementations. We elaborate on those differences below.

Computing threads: a thread is the smallest independent sequence of instructions in a computing process. CPU is able to handle complex tasks using a limited number of threads in the order of $O(10)$. In contrast, GPU has the capacity of running thousands to millions of threads in parallel, but each is capable of simpler operations. A GPU computing process is referred as *kernel*.

Sharing memory: comparing to single-thread tasks in CPU, memory access is much more involved under a multi-thread scenario like GPU. When more than one thread access a sharing memory location, a race condition can occur when they both attempt to modify the content, and their access order determines the outcome. As a result, the consistency and correctness of the results are not guaranteed without special treatments. To avoid a race condition, we should either explicitly enforce some order among the threads or avoid concurrent access to the same memory locations. The former option is not preferred as it undermines the benefits of parallelism. On the other hand, memory sharing restriction curbs inter-thread information exchange. As a result, an algorithm needs to avoid information exchange among its parallel components to achieve high performance on GPU.

Given the characteristics, although GPU has great potential to boost algorithms' performance through parallelization, a GPU-parallelized algorithm is subject to two kinds of communication overheads: (i) *CPU-GPU* and (ii) *thread-to-thread*. CPU-GPU communication incurs an overhead on copying large volume of data between the memory systems of CPU and GPU, and thread-to-thread communication imposes an overhead on handling coupled memory accesses among parallel threads.

GPU memory offers a limited resource to mitigate these overheads. In GPU we distinguish three types of memories:[1]

- Private memory. It is accessible to each of the individual single threads, access is fast but capacity is limited and information stored here cannot be shared with other threads
- Local memory. It is accessible to a *local* group of threads, and they can coordinate to write and read from these shared memory locations.
- Global memory. It is accessible to all threads, but they do not coordinate of reading and writing, so it cannot be used to share information among threads. It is slower than the other two.

Local memory offers a great potential to overcome coordination issues among threads, and the fact that communication is only local resembles the communication scheme in control systems for large-networks. Although a single thread cannot belong to two different groups (as opposed to a subsystem in a network structure, where groups–incoming and outgoing sets–overlap) the parallelization capabilities of GPU can be exploited to duplicate parts of the same subsystem and run them simultaneously. We will show that the DLMPC algorithm is especially well-suited to deal with them by virtue of its localized nature as it allows to take advantage of the GPU parallelization capabilities while respecting the communication constraints. Furthermore, we note how an explicit formulation to DLMPC is essential for such parallelization, since each GPU thread is limited to very simple operations.

GPU parallelization strategy of DLMPC

In this subsection, we provide the implementation of the DLMPC algorithm in a computer system equipped with both a CPU and a GPU. First, we take advantage of the parallelization potential of the algorithm and perform a *naive parallelization* of the ADMM steps in GPU. We then provide enhancements to reduce overheads, such as reducing setup complexity by using *longest-vector length*, and reducing CPU-GPU communication by setting *combined kernels*. Lastly, we present how the locality constraints allow to effectively use *local memory* to deal with the thread-to-thread coupling.

[1]We use OpenCL terminology. Although conceptually equivalent, terminology is slightly different from the one used in CUDA.

Throughout this subsection, we rely on graphical aids that capture the important features of the DLMPC computations such as Fig. 6.2 and Algorithms 6 and 7 to ease the exposition.

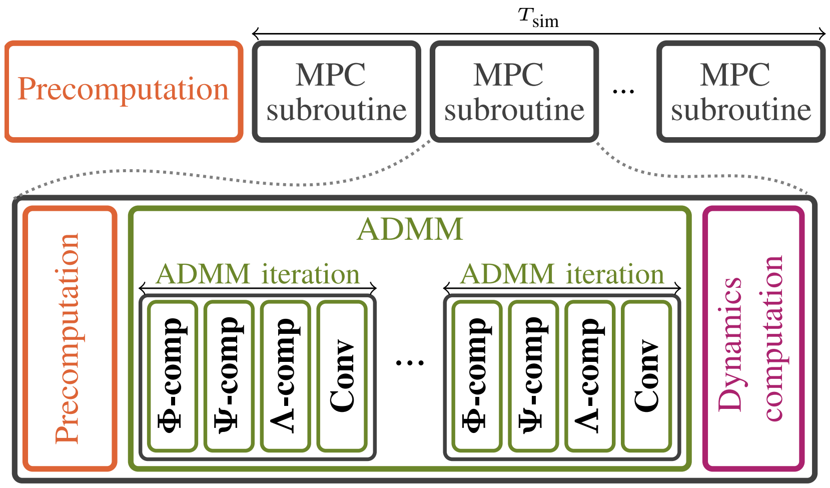

Figure 6.2: The DLMPC algorithm consists of different computation steps. A precomputation step is carried out to compute necessary matrices that stay constant throughout all MPC iterations. Once this is completed, MPC iterations run sequentially (one per time-step) for a given number T_{sim} of time steps. Within a MPC iteration, a precomputation step precedes the ADMM algorithm. Once converged, we compute the next control input and the state according to the dynamics. Each iteration of ADMM is composed by the steps detailed in Algorithm7.

Algorithm 6 Sketch of DLMPC implementation in GPU

```
1: Precompute necessities of closed-form solutions.
2: Initialize $x(0)$.
3: for $t = 0$ to simulation horizon ($T_{\text{sim}}$) do
4:     Precompute the explicit solution with Table 6.1 using $x_0 := x(t)$.
5:     Perform ADMM per Alg. 7.
6:     Compute $u(t)$ from $\mathbf{\Phi}$ and obtain $x(t+1)$.
7: end for
```

Algorithm 7 Sketch of ADMM computations for subsystem i

```
1: Conv ← false.
2: while Conv is false do
3:     Φ-comp: Compute rows of $\mathbf{\Phi}$ via the explicit solution in Lemma 16.
4:     Share $\mathbf{\Phi}$ with its $d$-hop local neighbors.
5:     Ψ-comp: Compute columns of $\mathbf{\Psi}$ with step 5 in Alg. 1.
6:     Share $\mathbf{\Psi}$ with its $d$-hop local neighbors.
7:     Λ-comp: $\mathbf{\Lambda} \leftarrow \mathbf{\Lambda} + \mathbf{\Phi} - \mathbf{\Psi}$.
8:     Share $\mathbf{\Lambda} f$ with its $d$-hop local neighbors.
9:     Conv: Check the convergence criterion and save the result in conv.
10: end while
```

Naive Parallelization

We start with a naive parallelization of DLMPC algorithm by parallelizing ADMM steps in 7. For each ADMM step, we assign each of the subproblems below a single thread in GPU:

- For **$\mathbf{\Phi}$-comp**, each thread computes one row of $\mathbf{\Phi}$.
- For **$\mathbf{\Psi}$-comp**, each thread computes one column of $\mathbf{\Psi}$.
- For **$\mathbf{\Lambda}$-comp**, each thread computes one element of $\mathbf{\Lambda}$.
- For **Conv**, each thread evaluates the convergence criterion against one column.

Notice that each thread in GPU is tasked with performing all necessary arithmetic operations leading to the assigned row/column/element. In this implementation we are not parallelizing the arithmetic computations in GPU, but rather treating each GPU thread as a subsystem of the distributed MPC framework. The reason for this choice will become apparent in the next subsection, and the additional computational overhead is small since computations are of small dimension because of the locality constraints.

According to the ADMM algorithm, abundant information sharing is required after each computation in order to perform the next computation. Due to the limitations of GPU communication among threads, in this naive parallelization we perform the information sharing in the form of memory accesses in CPU. Therefore, after each parallelized computation we return to CPU to exchange results and set up the next one. We illustrate this implementation in 6.3, where we represent the computing threads with an arrow so one can distinguish the steps that are computed in CPU (single thread) and the ones that are computed in GPU (multi-thread). Notice that an additional setup step required to launch the GPU kernels is also represented.

This naive parallelization of the DLMPC algorithm suffers from several overheads. First, the threads have different runtime due to the various lengths of the rows and columns they compute. Such various lengths result in significant setup overhead before computation. Second, there are several CPU-GPU switches per ADMM iteration to exchange information across different threads for rows and columns. This incurs CPU-GPU communication overhead. Those overheads imply that a naive parallelization of a distributed and localized MPC scheme such as DLMPC is not necessarily efficient, and additional considerations are needed to fully exploit the GPU potential. In what follows, we analyze these overheads and provide effective solutions based on hardware-specific considerations and the presence of locality constraints. We build upon these solutions until an optimal GPU implementation of the ADMM steps is presented.

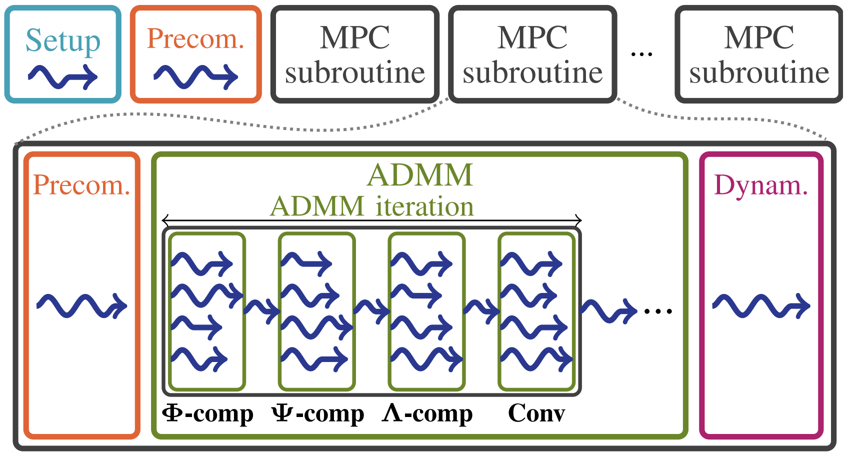

Figure 6.3: Naive parallelization implementation. Boxed components represent the same computation steps as in 6.2. An additional Setup step has been introduced at the beginning of the implementation for GPU setup. Computing threads are denoted with blue wavy arrows: A single arrow represents a single-thread computation, and multiple arrows within a computation step represent a multi-thread computation. The length of the different arrows in multi-thread processes represents runtime for each of the threads.

Longest-vector length

Threads have different runtime in 6.3 since they process different sizes of input vectors. Feeding each thread a different-sized input vector imposes a setup overhead–we need to compute, store, and pass as parameters of the threads the sizes of each input vector. Such an overhead was justifiable in a single thread CPU version like [59] as we want to avoid unnecessary computations. In particular, only non-zero elements are needed when computing on a single thread, and filtering out non-zero elements pays off as fewer inputs imply faster computation under sequential processing. The situation changes in GPU parallelization. Since the computation time of parallelized threads is determined by the slowest one, and the kernel does not return until *all* threads have finished the computations, it is no longer beneficial to trim off zero elements unless they are processed by the slowest thread.

Accordingly, we can save the efforts of attaining exact different-sized input vectors for parallelization. Instead, we only need to ensure the input vector is long enough to cover the non-zero elements and find the minimum upper bound on the length, which is the maximum number of non-zero elements in the rows and columns of $\mathbf{\Phi}$ and $\mathbf{\Psi}$, respectively, or the *longest-vector length* for short. We denote by D_{row} and D_{col} the longest-vector lengths of $\mathbf{\Phi}$ and $\mathbf{\Psi}$, respectively, and establish below that by virtue of the locality constraints, $D_{\text{row}}, D_{\text{col}} \ll N$ and the number of elements that a thread solves for is much smaller than the size of the network.

Lemma 17. *Let s be the maximum number of states or control inputs per subsystem in the network, and l the maximum degree of nodes in $\mathcal{G}_{(A,B)}$. Suppose $\mathcal{G}_{(A,B)}$ is subject to d-locality constraints and the MPC time horizon is T, then D_{row} and D_{col} are bounded by*

$$D_{row} \leq \frac{s(l^{d+1}-1)}{l-1}, \qquad D_{col} \leq \frac{(2T-1)s(l^{d+1}-1)}{l-1}.$$

Proof. Each row in $\mathbf{\Phi}$ represents a state/input in a subsystem, and hence D_{row} is the number of states that it can receive information from. We can establish a bound on D_{row} by bounding the number of nodes within d-hops. By definition, we know that there are at most l^k nodes that are k-hops away from a node, so within d-hops, there are at most

$$1 + l + l^2 + \cdots + l^d = \frac{l^{d+1}-1}{l-1}$$

nodes, each has at most s states, which shows the bound.

On the other hand, D_{col} is the number of states and inputs, among all horizon T, a state can impact. Similarly, by d-locality constraints, we can use the bound on D_{row} as an estimate of the states/inputs a state would impact at each time. Since there are, in total, T states and $T-1$ inputs in $\mathbf{\Psi}$ per column, we can bound D_{col} by $(2T-1)$ times of the above bound on D_{row}, which yields the desired result. □

Given a sparse system, the maximum degree l is expected to be small, as is the maximum number of subsystem states/inputs s. Given the assumption $d \ll N$, the above lemma suggests $D_{\text{row}}, D_{\text{col}} \ll N$

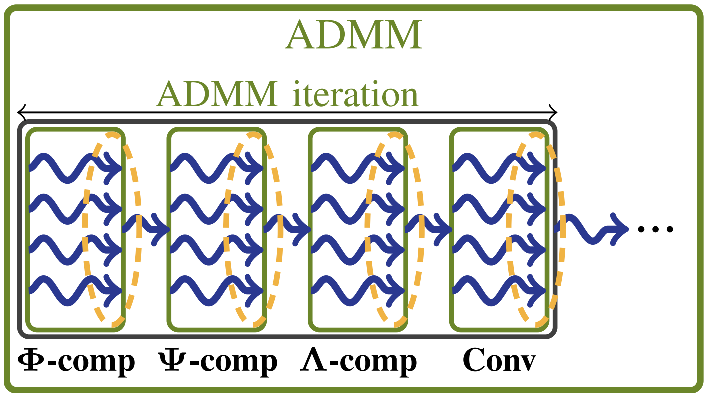

Figure 6.4: Since the computation time of parallelized threads is dominated by the slowest one, we could omit computing the exact input vector size for each thread and feed all threads with same-sized (the longest-sized) input vectors, which results in roughly the same runtime for all the threads. This simplifies both the setup and per-thread computation and hence reduces the overhead.

Combined kernels

We then tackle the CPU-GPU communication overhead. In the naive parallelization, CPU-GPU communication are necessary to properly exchange information in between computations. The reason is that each computation occurs according to a different distribution of the elements of $\mathbf{\Phi}$, $\mathbf{\Psi}$, $\mathbf{\Lambda}$, i.e., row-wise, column-wise, and element-wise. Although this particular distribution of elements might be the most efficient for each of the computations isolated, the additional GPU-CPU overhead steaming from the information sharing in between computations makes this option is suboptimal.

To reduce the CPU-GPU communication overhead, we proposed the use of combined kernels. In particular, the last three computation steps in the ADMM iteration can be combined in the same kernel by parallelizing **$\mathbf{\Lambda}$-comp** in a column-wise fashion (as opposed to element-wise). By distributing the computations in this manner, each of the threads has sufficient information to sequentially perform **$\mathbf{\Psi}$-comp**, **$\mathbf{\Lambda}$-comp**, and **Conv** without the need for communication among threads. This reduces the CPU-GPU communication overhead since only one exchange between CPU and GPU is necessary for each ADMM iteration (for the transformation from row-wise to column-wise). However, the treatment herein slightly degrades the parallelization benefits of **$\boldsymbol{\Lambda}$-comp**, since by having only a thread per column, each thread now has to loop over the elements in its column sequentially. This additional overhead is very modest because due to the locality constraints, the number of relevant elements per thread is D_{col}, as opposed to a CPU-GPU memory-copying operation, where the variables handled are of order N.

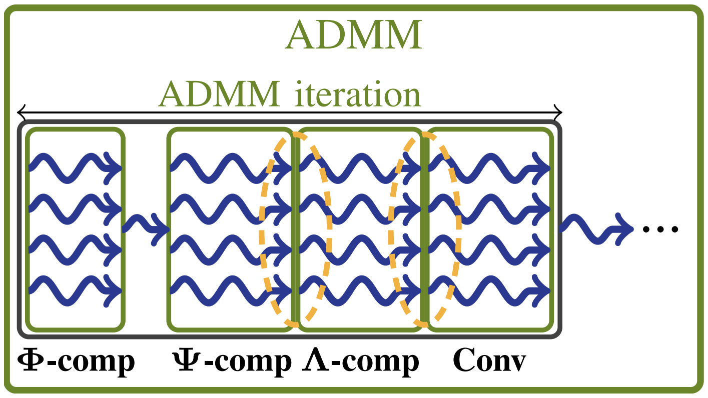

Figure 6.5: To reduce CPU-GPU communication overhead, we remove the single-thread computation in between computations (circled in yellow) by combining the computations into per-column threads. This enhances the parallelization by combining kernels. Such a combination does not apply to the information exchange between **$\boldsymbol{\Phi}$-comp** and **$\boldsymbol{\Psi}$-comp** as we shift from per-row to per-column computation.

Local memory and column patch

The information exchange between **$\boldsymbol{\Phi}$-comp** and **$\boldsymbol{\Psi}$-comp** involves the transformation from row-wise to column-wise representation and hence is not easily combined into one kernel. The key difficulty is that the row-wise results should be passed down to per-column threads, which results in thread-to-thread communication/synchronization issues. We could realize synchronized thread-to-thread communications through local memory. However, local memory is shallow–it would not fit all the threads in–and it is exclusive–one thread can only belong to one group. These properties make the bipartite information exchange pattern difficult to enforce: The row-wise result might be required by multiple per-column threads, while each thread may need multiple row-wise results. As a result, to leverage local memory to save CPU-GPU communication overhead, we need to group the threads smartly.

Our approach is to group each per-column thread in **$\boldsymbol{\Psi}$-comp** with row-wise computation threads in **$\boldsymbol{\Phi}$-comp**, referred to as *column patch*. That is, for the i^{th} column of $\boldsymbol{\Psi}$, we launch a column patch to solve for the j^{th} row where $j \in \{j : \boldsymbol{\Phi}(j,i) \neq 0\}$.

Once this is done, the row-wise results are saved in local memory in GPU, and one of those threads can proceed with the column-wise computations of $\mathbf{\Psi}$ and $\mathbf{\Lambda}$ as described in the previous subsection without returning to CPU.

Thanks to the locality constraints, each column patch only has D_{col} **$\mathbf{\Phi}$-comp** threads to include and fit their results in the shallow local memory. On the other hand, since each row has several non-zero elements, we could potentially have multiple threads in different column patches that compute the same row-wise result. But it is fine as GPU has plenty of threads to launch, and using multiple threads to compute the same result in parallel does not incur additional runtime overhead. Therefore, we ensure synchronization without the need for information sharing across threads - since we can repeat relevant computations in different local groups - or computing units - since local synchronization is all is needed. This was only possible by exploiting the GPU architecture together with the locality constraints directly encoded in the DLMPC formulation.

We highlight the roles of the locality constraints in our enhancement techniques. Locality constraints can facilitate desirable trade-offs between computational resources and information exchange across threads: Longest-vector length incurs additional precomputation steps, and combined kernels sacrifice parallelization potential of **$\mathbf{\Lambda}$-comp**. Those trade-offs are justified by the small dimensions derived from the locality constraints $D_{\text{row}}, D_{\text{col}} \ll N$. Meanwhile, for the local memory and column patch technique, locality constraints allow us to decouple the row-wise threads without harming the runtime (in essence, locality allow us to pay in the spatial space to decouple the row-wise threads without temporal performance degradation).

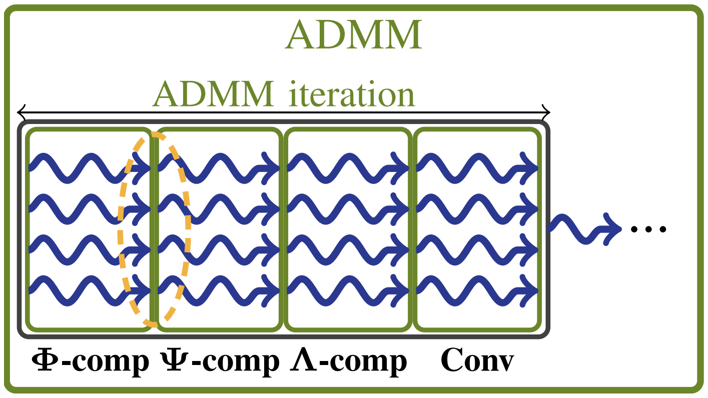

Figure 6.6: Conducting **Φ-comp** and **Ψ-comp** consecutively in GPU requires the per-row threads to exchange information with per-column threads, which results in thread-to-thread communication. Using local memory, we can avoid such a thread-to-thread communication by creating column patches, which duplicate row-wise computation threads to decouple the per-column local memory groups and synchronize results entirely within GPU.

6.5 Simulation Experiments

Through simulations, we study two aspects of the GPU-parallelized DLMPC. First, we compare the scalability of our implementation with other methods. We then analyze the overhead of the implementations for future enhancements.

Setup

We implement our GPU-parallelized DLMPC in Python and OpenCL. We compare the scalability of the four proposed GPU implementations from the previous section against two CPU variations–a Python replica of the single-threaded DLMPC version in [59] and an optimization-based approach under the SLSpy framework [106] with CVXPY [107] as the solver. The results are measured on a desktop with AMD Ryzen 7 3700X processor (16 logical cores), 32 GB DDR4 memory, and AMD Radeon RX 550/550X GPU. For each evaluated scenario, we simulate 100 different initial conditions and present the average and the standard deviation of the measurements. The synthetic dynamics is chosen are a chain-like network with two-state nodes as the subsystems. Each subsystem i evolves according to

$$[x(t+1)]_i = [A]_{ii}[x(t)]_i + \sum_{j \in \mathbf{in}_i(d)} [A]_{ij}[x(t)]_j + [B]_{ii}[u(t)]_i,$$

where $\mathbf{in}_i(d)$ contain the d-hop neighbors of node i and

$$[A]_{ii} = \begin{bmatrix} 1 & 0.1 \\ -0.3 & 0.7 \end{bmatrix}, \ [A]_{ij} = \begin{bmatrix} 0 & 0 \\ 0.1 & 0.1 \end{bmatrix}, \ [B]_{ii} = I.$$

The state is subject to upper and lower bounds:

$$-0.2 \leq [x(t)]_{i,1} \leq 1.2 \quad \text{for } t = 1, ..., T,$$

where $[x]_{i,1}$ is the first state in the two-state subsystem i.

We perform the MPC with $T_{\text{sim}} = 20$ subroutine iterations with the cost function

$$f(x,u) = \sum_{i=1}^{N} \sum_{t=1}^{T-1} \|[x(t)]_i\|_2^2 + \|[u(t)]_i\|_2^2 + \|[x(T)]_i\|_2^2.$$

Note that the number of states and inputs in this plant is $3N$, since each of the N subsystems has 2 states and 1 input.

Scalability

To evaluate the scalability of the methods, we run the simulations with varying system size N, MPC time horizon T, and locality region size d. We measure the average runtime per MPC iteration, i.e., the total runtime divided by the number of MPC subroutine iterations T_{sim}, and summarize the results in 6.7. We remark that the runtime is measured for the *whole* simulation rather than merely the ADMM portion of the algorithm.

In 6.7, the GPU implementations scale much better with the network size N than the CPU implementations. Moreover, the runtime differences grow from an order of magnitude to several orders of magnitude as N increases. This is as expected since GPU implementations can parallelize the subsystem computations through multi-threads whereas the CPU implementations cannot. Remarkably, the ADMM implementation in CPU is consistently worse than when solved via CVXPY, which emphasizes the need for a parallel implementation such as the one presented in this paper to fully take advantage of the DLMPC algorithm. Among the GPU methods, local memory has superior performance (50× faster than ADMM in CPU for $N = 25$ and 35× for $N = 100$), followed by combined kernels and longest-vector length (15× faster than ADMM in CPU for $N = 25$ and 25× for $N = 100$). Naive parallelization is 3× slower than local memory for small N. In fact, a smaller N leads to a bigger performance difference among GPU implementations, which indicates that the CPU overhead of the implementations outweighs the improvements made by GPU when the network scales.

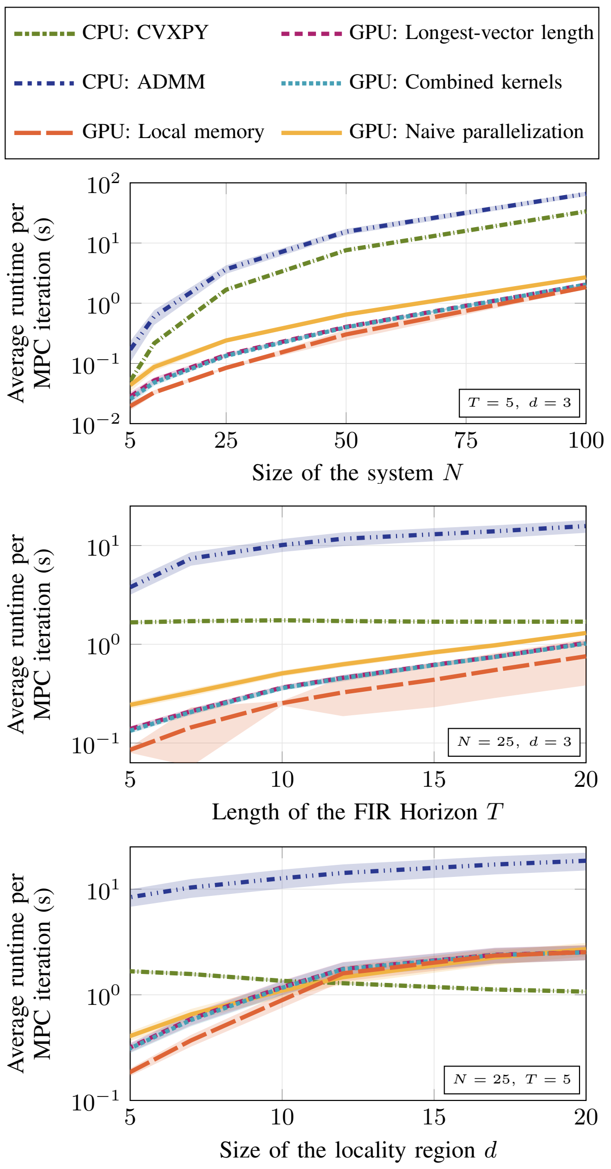

Figure 6.7: Comparison of the runtimes obtained by different computing strategies for different parameter regimes. The lines are the mean values and the shaded areas show the values within one standard deviation. We observe that GPU computation strategies scale much better with the size of the network than CPU implementations. This trend also holds for ADMM CPU implementation over all time horizons and locality region sizes, while GPU implementations only outperform CVXPY on small locality regions and CVXPY seems to scale well over all simulated time horizons.

For time horizon T and locality region size d, the runtime scales accordingly for all the ADMM implementations. This can be seen from a simple analysis of the optimization variables: Since larger T and d introduce more non-zero elements in the matrix $\mathbf{\Phi}$, the corresponding decomposed row and column vectors become

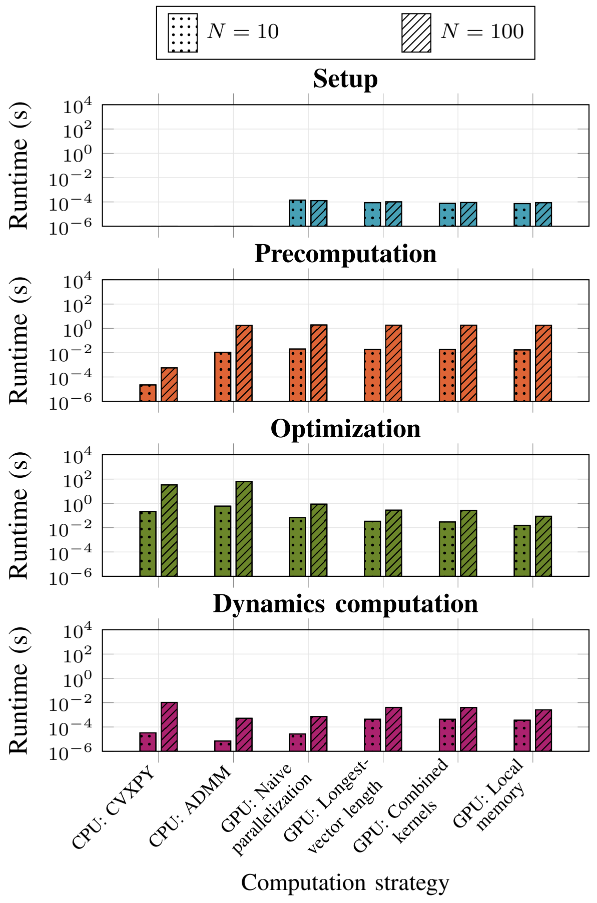

Figure 6.8: Runtime breakdown for the different computing phases: precompilation, precomputation, optimization and dynamics computation for two different network sizes $N = 10$ and $N = 100$. Colors for the different phases are the same as in 6.3. For CPU implementations, the optimization phase is the bottleneck, while the bottleneck shifts to the precomputation for the GPU implementations.

longer and the runtime increases. However, this trend does not apply to the SLSpy-based CPU implementation solved by CVXPY, where the runtime stays pretty much the same or even decreases when T and d increases. Nevertheless, the GPU implementations still outperform CVXPY by up to one order of magnitude for $d \leq 10$ and all simulated T. In other words, the advantage of GPU parallelization is significant when the locality region is small. Hence, in light of the results, the ADMM implementation of DLMPC is best for large network consisting of subsystems with relatively small local neighborhood.

Overhead analysis

To better understand how different methods scale in runtime with the network size, we break down the runtime into four DLMPC phases as in 6.3, which are:

Setup refers to all the computational steps necessary to setup the GPU computations. It refers to the compilation of the kernels themselves, but also to all the extra computations necessary to implement the different GPU schemes, such as the computation of D_{row} and D_{col}, etc.

Precomputation refers to computing the necessary matrices and vectors that multiply the decision variables $\mathbf{\Phi}$, $\mathbf{\Psi}$, $\mathbf{\Lambda}$. We note here that this phase appears twice in the implementation: before starting any of the MPC computations, and before starting each of the MPC subroutines.

Optimization refers to the computational steps necessary to solve the optimization problem (6.3a). In this case, it refers to the steps taken by ADMM or CVXPY to find a value for $\mathbf{\Phi}$.

Dynamics computation refers to the computation of the next state using the control input given by MPC.

We measure the runtime breakdown for network size $N = 10$ and $N = 100$ and present the results in 6.8. From 6.8, we can observe that the setup and dynamics computation phases are relatively fast, and the runtime is dominated by the optimization phase when $N = 10$ and the precomputation phase when $N = 100$. As we adopt more sophisticated techniques described in this chapter to GPU implementations, the optimization phase speeds up significantly. However, the precomputation phase does not benefit from the techniques and it becomes the bottleneck when the network size scales. Since an increase in d or T mostly burdens the setup and precomputation phases, the results also explain why the GPU implementations scale poorer when d or T is large.

Another important takeway from 6.8 is that for CPU implementations, the optimization phase is the bottleneck, while the bottleneck shifts to the precomputation for the GPU implementations. This justifies our approaches in this paper to accelerate the CPU optimization phase by GPU parallelization. Meanwhile, the results also suggest that to further speed up the computation, future research should focus on faster precomputation techniques.

6.6 Conclusion and Future work

We introduced an explicit solution to the MPC problem that can be applied to large networked systems. Inspired by [48], our explicit solution partitions the space into three regions per state/input instantiation, and with the assumptions that no coupling between states is allowed, each subsystem can solve for each state (input) instantiation sequentially or in parallel, which results in a fast computation runtime per subsystem. Since each subsystem runs its own optimization problem in parallel, this results in large runtime improvements. Computational experiments show that the runtime of each MPC iteration per subsystem in the network scales with $O(1)$ complexity as the size of the network increases.

We used these results develop effective GPU parallelized DLMPC for large-scale distributed system control. Our results show that although a naive GPU implementation does improve the performance by $15-25\times$, we can still get up to $50\times$ performance improvement by taking into account the hardware-intrinsic limitations. We overcome these limitations by taking advantage of the local communication constraints in the formulation, and developing longest-vector length, combined kernels and local memory implementations. With extensive experiments, we demonstrate that the DLMPC algorithm is suitable for GPU parallelization, and that its full potential is only realized when the local communication constraints are taken into account in the GPU implementation. We demonstrate the scalability of the method for large network sizes, and noticed that most of the computational overhead in the GPU computations was due to the precomputations being performed in CPU.

As discussed in the overhead analysis, precomputation becomes the new overhead after our GPU parallelizations. Therefore, a future direction would be to effectively parallelize the precomputation for higher performance. In addition, there are some other parts of DLMPC that we can parallelize further, such as a better initial point for ADMM to converge faster as well as a better parallelized dynamics computation. Another future direction is to include more than one ADMM iteration into one kernel, to avoid CPU-GPU data exchanges. One might also be interested in extending the parallelization in this work to a fully distributed setting where the processing units scatter over a network. As the distributed setting introduces new challenges such as robustness to communication dropouts, synchronization, and delay, it would be interesting to see how locality constraints could improve robustness or performance.

Chapter 7

CONCLUSION

7.1 Challenge Addressed

Distributed control of large-scale systems presents significant challenges when relying on a centralized controller. The computational demands of Model Predictive Control (MPC) often make it impractical to implement a centralized approach. However, adapting centralized MPC ideas to the distributed setting is a complex task. Contemporary large-scale distributed systems like the Internet of Things benefit from widespread sensing and communication capabilities but face local resource constraints in power consumption, memory, and computational power. To transform these systems from passive data collectors to active distributed control systems, it is essential to develop algorithmic approaches that exploit the inherent advantages while considering the network's underlying resource limitations.

There have been substantial efforts in the literature towards a distributed MPC approach. However, addressing the challenges of distributed MPC requires more than simply extending centralized approaches. With this thesis, we aim at establishing a framework that leverages the inherent network and communication structure, enabling scalable algorithms that are independent of the global system's size. Our goal is to achieve this in a computationally efficient and non-conservative manner, making DMPC applicable in real-world scenarios. To achieve this, additional considerations are necessary to provide efficient and non-conservative theoretical guarantees, while also thoroughly understanding the impact of local communication constraints on performance. Furthermore, we believe that adopting a data-driven approach to DMPC is crucial as it eliminates the need for costly system identification algorithms. As we have shown, for such systems, collecting local trajectory data from a subset of neighboring systems is a more feasible alternative to developing intricate system models required by model-based control algorithms. At present, this thesis is–to the best of our knowledge– the first comprehensive framework for DMPC that incorporates the underlying system structure and offers efficient, scalable, data-driven, and non-conservative solutions with theoretical guarantees.

7.2 Main Contributions

The main contribution of this thesis is a novel optimal and robust control framework based on MPC that is able to achieve stringent requirements with highly-scalable communications and computing. We show how these results extend naturally to the data-driven case where no models are available and control is based on past observations only. We also provide novel hardware implementations that exploit GPU technology to further accelerate computations. In order to achieve this, we take advantage of the sparsity of the underlying systems by means of introducing a new type of constraints in the formulation: locality constraints restrict each subsystem in the network to only communicate and influence a small neighborhood of subsystems as opposed to the entire network. This gives rise to a new set of algorithms, theoretical results and architectures to optimally control distributed cyber-physical systems for safety-critical applications. In what follows we detail our main contributions.

Distributed and localized synthesis and implementation of closed-loop model predictive controllers (MPC)

We present a novel MPC algorithm designed for large-scale linear systems, offering both synthesis and implementation capabilities in a distributed and localized manner, without relying on strong assumptions. Our algorithm, called Distributed and Localized Model Predictive Control (DLMPC), enables the computation and implementation of control actions by exchanging only local state and model information between subsystems. Notably, the resulting distributed algorithms demonstrate robustness against various types of additive disturbances, and their computational scalability is independent of the network size for the first time. This significant advancement makes our approach scalable for systems of arbitrary sizes.

Minimally conservative guarantees for asymptotic stability and recursive feasibility

Previous works have faced challenges when introducing guarantees into their algorithms due to the distributed nature of the problem. These guarantees often led to excessive conservatism, compromising the solutions provided, or added computational burden while still introducing some level of conservatism. In this thesis, we address these issues by presenting theoretical results and algorithms that compute theoretical guarantees for stability and feasibility of MPC. Importantly, these computations can be performed offline, prior to implementing the DLMPC algorithm, which ensures they do not contribute to the computational burden. Moreover, we do so while introducing minimal conservatism. Notably, due to the constraints imposed by locality, these computations can now be performed in a distributed and localized manner for the first time, resulting in highly scalable computations.

Globally optimal guarantees as compared with global MPC

We conduct a thorough analysis comparing the optimal performance of DLMP with a MPC scheme that permits global communication. Our findings indicate that when the system's underlying topology is sparse, as is commonly observed in large-scale networks, the inclusion of local communication constraints does not lead to suboptimal solutions. These results emphasize the benefits of introducing locality constraints. Despite not affecting global optimality in many scenarios, these constraints offer a more scalable and efficient solution compared to the global counterpart.

Data-driven extension

We extend these results to the purely data-driven scenario in the noiseless case, eliminating the need for a system model. Remarkably, we demonstrate that all the previous guarantees still hold, and the requirement for a model is entirely replaced by past-trajectory data. Additionally, thanks to the locality constraints, the resulting algorithm maintains a small sample complexity that remains independent of the overall network size. This contribution is particularly valuable for real-world systems, as they often possess large dimensions and extreme complexity, making it infeasible to have an accurate model of the system. The data-driven version of DLMPC brings the benefits of optimal control with guarantees within reach for these systems, which are often missing in machine learning appraoches.

Efficient hardware implementation

Lastly, we develop an efficient GPU implementation of DLMPC by capitalizing on the favorable structure of the DLMPC problem that aligns well with the capabilities of GPU computations. This contribution highlights that the advantages of locality constraints extend beyond distributed settings, proving valuable even in centralized scenarios that demand parallelizable and efficient computations. The findings of this work demonstrate an additional benefit of locality constraints, enabling the creation of layered and efficient control architectures. By leveraging the power of GPUs, we unlock new possibilities for enhancing the performance and scalability of DLMPC, further highlighting the utility of locality constraints for control applications.

In summary, this thesis introduces a comprehensive framework for online distributed control algorithms under constraints. The key contribution is the development of DLMPC, an online distributed control algorithm that achieves robustness, scalability, efficiency, and data-driven computations while maintaining theoretical guarantees and supporting parallel hardware implementations. The framework emphasizes the significance of considering locality constraints and opens up new possibilities for the development of layered control systems empowered by these constraints. The theoretical contributions of this work address several longstanding challenges in the field of distributed control, providing solutions to complex theoretical problems. Furthermore, the extension to a data-driven approach enables the application of these results to real-world distributed systems that lack a precise system model. Additionally, the hardware implementation aspect offers the potential to leverage the proposed approach even in centralized settings, significantly accelerating computations. By combining these different frontiers, effective and scalable control architectures can be created for complex and large-scale real-world systems. This thesis showcases the achievement of such goals by harnessing the intrinsic and widespread sparsity observed in technology and biological large-scale distributed systems.

7.3 Future Research Directions

Besides the immediate research directions outlined at the end of each chapter, in what follows we propose longer-term research directions that will expand the work of this thesis.

Communication exchanges

Most algorithms in the DLMPC framework rely on multiple exchanges of information between the subsystems. For this reason, a better analysis of how communication loss affects the closed-loop performance of the algorithm is an interesting question. Although a formal analysis is in order, the work done in [62] suggests that it would be possible to slightly modify the proposed ADMM-based scheme to make it robust to unreliable communication links.

Furthermore, extending these results to information exchange topologies with sparsity and delays is an interesting research direction. While the SLS framework handles delays in the implementation structure of a distributed controller, incorporating such constraints into a distributed synthesis problem poses challenges. Developing algorithms that consider delays in information exchange while preserving performance and theoretical guarantees is a promising avenue for future research. Addressing these challenges would enhance the applicability of distributed control algorithms in real-world scenarios, contributing to the creation of robust control architectures for large-scale systems.

Robust data-driven formulation

This thesis explores two types of noise distributions within the model-based approach. However, there is an open question regarding the exploration of these results in the data-driven case and their implications for theoretical guarantees. Furthermore, it would be valuable to investigate the efficient sample complexity required to effectively solve the problem in this context. Along these lines, an interesting research direction would be to explore different cost formulations and objective functions that can mitigate the conservatism of the formulation. Such investigations have the potential to enhance the overall performance by providing more efficient and scalable solutions, enhancing the applicability of DLMPC to a larger range of settings where only past trajectories are available and robustness to disturbances is needed.

Layered MPC architectures

MPC, despite its online nature, is known to be slower compared to offline controllers. However, its unique ability to handle constraints in real-time allows for less conservative control by more effectively handling worst-case scenarios. To effectively control complex large-scale systems, it has become evident that layering control strategies holds promise (see the work in [108] for details). However, the question of how to layer and interface different controllers remains open.

While DLMPC stands out for providing optimal and scalable algorithms with guarantees, its role within a layered architecture and the extent of control it should handle in conjunction with other layers are not yet clear. Furthermore, the optimal architecture and functions of these controllers in a layered setup are still uncertain. Further exploration of these ideas is crucial to develop effective, optimal, efficient, and scalable controllers for large-scale distributed systems.

References

[1] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017. DOI: 10.1017/9781139061759.

[2] D. Q. Mayne, M. M. Seron, and S. V. Rakovic, "Robust model predictive control of constrained linear systems with bounded disturbances," *Automatica*, vol. 41, pp. 219–224, 2005. DOI: 10.1016/j.automatica.2004.08.019.

[3] A. Venkat, I. Hiskens, J. Rawlings, and S. Wright, "Distributed MPC strategies with application to power system automatic generation control," *IEEE Trans. Control Syst. Tech.*, vol. 16, no. 6, pp. 1192–1206, 2008. DOI: 10.1109/TCST.2008.919414.

[4] Y. Zheng, S. Li, and H. Qiu, "Networked coordination-based distributed model predictive control for large-scale system," *IEEE Trans. Control Syst. Tech.*, vol. 21, no. 3, pp. 991–998, 2013. DOI: 10.1109/TCST.2012.2196280.

[5] P. Giselsson, M. D. Doan, T. Keviczky, B. D. Schutter, and A. Rantzer, "Accelerated gradient methods and dual decomposition in distributed model predictive control," *Automatica*, vol. 49, no. 3, pp. 829–833, 2013. DOI: 10.1016/j.automatica.2013.01.009.

[6] 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*, vol. 69, pp. 117–125, 2016. DOI: 10.1016/j.automatica.2016.02.009.

[7] R. E. Jalal and B. P. Rasmussen, "Limited-communication distributed model predictive control for coupled and constrained subsystems," *IEEE Trans. Control Syst. Tech.*, vol. 25, no. 5, pp. 1807–1815, 2017. DOI: 10.1109/TCST.2016.2615088.

[8] Z. Wang and C.-J. Ong, "Distributed MPC of constrained linear systems with online decoupling of the terminal constraint," in *Proc. IEEE ACC*, 2015, pp. 2942–2947. DOI: 10.1109/ACC.2015.7171182.

[9] A. N. Venkat, J. B. Rawlings, and S. J. Wright, "Stability and optimality of distributed model predictive control," in *Proc. IEEE CDC*, 2005, pp. 6680–6685. DOI: 10.1109/CDC.2005.1583235.

[10] Y. R. Sturz, E. L. Zhu, U. Rosolia, K. H. Johansson, and F. Borrelli, "Distributed learning model predictive control for linear systems," in *Proc. IEEE CDC*, 2020, pp. 4366–4373. DOI: 10.1109/CDC42340.2020.9303820.

[11] G. Grimm, M. J. Messina, S. E. Tuna, and A. R. Teel, "Examples when nonlinear model predictive control is nonrobust," *Automatica*, vol. 40, no. 10, pp. 1729–1738, 2004. DOI: 10.1016/j.automatica.2004.04.014.

[12] C. Conte, M. N. Zeilinger, M. Morari, and C. N. Jones, "Robust distributed model predictive control of linear systems," in *Proc. IEEE ECC*, 2013, pp. 2764–2769. DOI: 10.23919/ECC.2013.6669745.

[13] V. Blondel and J. N. Tsitsiklis, "NP-hardness of some linear control design problems," *SIAM J. Control Optim.*, vol. 35, no. 6, pp. 2118–2127, 1997. DOI: 10.1137/S036301299427263.

[14] A. Richards and J. P. How, "Robust distributed model predictive control," *Int. J. Control*, vol. 80, no. 9, pp. 1517–1531, 2007. DOI: 10.1016/j.automatica.2021.110141.

[15] P. J. Goulart, E. C. Kerrigan, and J. M. Maciejowski, "Optimization over state feedback policies for robust control with constraints," *Automatica*, vol. 42, no. 4, pp. 523–533, 2006. DOI: 10.1016/j.automatica.2005.08.023.

[16] M. Rotkowitz and S. Lall, "A characterization of convex problems in decentralized control," *IEEE Trans. Autom. Control*, vol. 51, no. 2, pp. 274–286, 2006. DOI: 10.1109/TAC.2005.860365.

[17] L. Furieri and M. Kamgarpour, "Robust control of constrained systems given an information structure," in *Proc. IEEE CDC*, Melbourne, Australia, 2017, pp. 3481–3486. DOI: 10.1109/CDC.2017.8264169.

[18] C. Langbort, C. R.S., and D. R., "Distributed control design for systems interconnected over an arbitrary graph," *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1502–1519, 2004. DOI: 10.1109/TAC.2004.834123.

[19] A. Jokic and M. Lazar, "On decentralized stabilization of discrete-time nonlinear systems," in *Proc. IEEE ACC*, 2009, pp. 5777–5782.

[20] A. I. Zecevic and D. D. Siljak, *Control of Complex Systems*. New York: Commun. Control Eng., Springer, 1988. DOI: 10.1016/C2015-0-02422-4.

[21] B. T. Stewart, A. Venkat, J. Rawlings, S. Wright, and G. Pannocchia, "Cooperative distributed model predictive control," *Syst. Control Lett.*, vol. 59, no. 8, pp. 460–469, 2010. DOI: 0.1016/j.sysconle.2010.06.005.

[22] J. M. Maestre, D. Muñoz de la Peña, E. F. Camacho, and T. Alamo, "Distributed model predictive control based on agent negotiation," *J. Process Control*, vol. 21, no. 5, pp. 685–697, 2011. DOI: 10.1016/j.jprocont.2010.12.006.

[23] P. A. Trodden and J. M. Maestre, "Distributed predictive control with minimization of mutual disturbances," *Automatica*, vol. 77, pp. 31–43, 2017. DOI: 10.1016/j.automatica.2016.11.023.

[24] G. Darivianakis, A. Eichler, and J. Lygeros, "Distributed model predictive control for linear systems with adaptive terminal sets," *IEEE Trans. Autom. Control*, vol. 65, no. 3, pp. 1044–1056, 2020. DOI: 10.1109/TAC.2019.2916774.

[25] A. Aboudonia, A. Eichler, and J. Lygeros, “Distributed model predictive control with asymmetric adaptive terminal sets for the regulation of large-scale systems,” *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6899–6904, 2020, 21st IFAC World Congress. DOI: 10.1016/j.ifacol.2020.12.356.

[26] S. Muntwiler, K. P. Wabersich, A. Carron, and M. N. Zeilinger, “Distributed model predictive safety certification for learning-based control,” *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 5258–5265, 2020. DOI: 10.1016/j.ifacol.2020.12.1205.

[27] Y. Wang and C. Manzie, “Robust distributed model predictive control of linear systems: Analysis and synthesis,” *Automatica*, vol. 137, p. 110 141, 2022. DOI: 10.1016/j.automatica.2021.110141.

[28] A. Gasparri and A. Marino, “A distributed framework for kk-hop control strategies in large-scale networks based on local interactions,” *IEEE Transactions on Automatic Control*, vol. 65, no. 5, pp. 1825–1840, 2020. DOI: 10.1109/TAC.2019.2926595.

[29] L. Ballotta and V. Gupta, “Faster consensus via a sparser controller,” *IEEE Control Systems Letters*, vol. 7, pp. 1459–1464, 2023. DOI: 10.1109/LCSYS.2023.3268005.

[30] J. Jiao, H. L. Trentelman, and M. K. Camlibel, “Distributed linear quadratic optimal control: Compute locally and act globally,” *IEEE Control Systems Letters*, vol. 4, no. 1, pp. 67–72, 2020. DOI: 10.1109/LCSYS.2019.2922189.

[31] S. Shin, Y. Lin, G. Qu, A. Wierman, and M. Anitescu, “Near-optimal distributed linear-quadratic regulator for networked systems,” *arXiv preprint arXiv:2204.05551*, 2022. DOI: 10.48550/ARXIV.2204.05551.

[32] H. K. Mousavi and N. Motee, “Explicit characterization of performance of a class of networked linear control systems,” *IEEE Transactions on Control of Network Systems*, vol. 7, no. 4, pp. 1688–1699, 2020. DOI: 10.1109/TCNS.2020.2995825.

[33] W. Tang and P. Daoutidis, “The role of community structures in sparse feedback control,” in *Proc. IEEE ACC*, 2018, pp. 1790–1795. DOI: 10.23919/ACC.2018.8431002.

[34] J. S. Baras and P. Hovareshti, “Effects of topology in networked systems: Stochastic methods and small worlds,” in *Proc. IEEE CDC*, 2008, pp. 2973–2978. DOI: 10.1109/CDC.2008.4738895.

[35] S. Lucia, M. Kögel, P. Zometa, D. Quevedo, and R. Findeisen, “Predictive control in the era of networked control and communication - A perspective,” *IFAC-PapersOnLine*, vol. 48, no. 23, pp. 322–331, 2015. DOI: 10.1016/j.ifacol.2015.11.302.

[36] A. Bemporad and D. Barcelli, "Decentralized model predictive control," in *Networked Control Systems*, A. Bemporad, M. Heemels, and M. Johansson, Eds. London: Springer London, 2010, pp. 149–178. DOI: 10.1007/978-0-85729-033-5_5.

[37] H. Hjalmarsson, M. Gevers, S. Gunnarsson, and O. Lequin, "Iterative feedback tuning: Theory and applications," *IEEE Control Systems Magazine*, vol. 18, no. 4, pp. 26–41, 1998. DOI: 10.1109/37.710876.

[38] M. Fazel, R. Ge, S. Kakade, and M. Mesbahi, "Global convergence of policy gradient methods for the linear quadratic regulator," in *Int. Conf. Mach. Learn.*, PMLR, 2018, pp. 1467–1476.

[39] H. Mohammadi, M. Soltanolkotabi, and M. R. Jovanović, "On the linear convergence of random search for discrete-time LQR," *IEEE Control Syst. Lett.*, vol. 5, no. 3, pp. 989–994, 2020. DOI: 10.1109/LCSYS.2020.3006256.

[40] S. Bradtke, B. Ydstie, and A. Barto, "Adaptive linear quadratic control using policy iteration," in *Proceedings of 1994 American Control Conference - ACC '94*, vol. 3, 1994, 3475–3479 vol.3. DOI: 10.1109/ACC.1994.735224.

[41] C. De Persis and P. Tesi, "Formulas for data-driven control: Stabilization, optimality, and robustness," *IEEE Trans. Autom. Control*, vol. 65, no. 3, pp. 909–924, 2019. DOI: 10.1109/TAC.2019.2959924.

[42] H. L. Trentelman, H. J. van Waarde, and M. K. Camlibel, "An informativity approach to data-driven tracking and regulation," *arXiv preprint arXiv:2009.01552*, 2020. DOI: 10.48550/ARXIV.2009.01552.

[43] F. Dörfler, J. Coulson, and I. Markovsky, "Bridging direct and indirect data-driven control formulations via regularizations and relaxations," *IEEE Transactions on Automatic Control*, vol. 68, no. 2, pp. 883–897, 2023. DOI: 10.1109/TAC.2022.3148374.

[44] J. Coulson, J. Lygeros, and F. Dörfler, "Distributionally robust chance constrained data-enabled predictive control," *IEEE Transactions on Automatic Control*, vol. 67, no. 7, pp. 3289–3304, 2022. DOI: 10.1109/TAC.2021.3097706.

[45] J. Berberich, J. Köhler, M. A. Müller, and F. Allgöwer, "Data-driven tracking MPC for changing setpoints," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6923–6930, 2020. DOI: 10.1016/j.ifacol.2020.12.389.

[46] A. Xue and N. Matni, "Data-driven system level synthesis," in *Proc. Mach. Learn. Research*, vol. 144, 2021, pp. 1–12.

[47] S. Alemzadeh, S. Talebi, and M. Mesbahi, "D3pi: Data-driven distributed policy iteration for homogeneous interconnected systems," *arXiv preprint arXiv:2103.11572*, 2021. DOI: 10.48550/ARXIV.2103.11572.

[48] A. Bemporad, M. Morari, V. Dua, and E. Pistikopoulos, "The explicit solution of model predictive control via multiparametric quadratic programming," in *Proceedings of the 2000 American Control Conference. ACC*, IEEE, 2000, 872–876 vol.2. DOI: 10.1109/ACC.2000.876624.

[49] A. Bemporad, N. A. Bozinis, V. Dua, M. Morari, and E. N. Pistikopoulos, "Model predictive control: A multi-parametric programming approach," in *Computer Aided Chemical Engineering*, vol. 8, Elsevier, 2000, pp. 301–306. DOI: 10.1016/S1570-7946(00)80052-8.

[50] E. N. Pistikopoulos, V. Dua, N. A. Bozinis, A. Bemporad, and M. Morari, "On-line optimization via off-line parametric optimization tools," *Computers and Chemical Engineering*, p. 11, 2002. DOI: 10.1016/S0098-1354(01)00739-6.

[51] J. Anderson, J. C. Doyle, S. H. Low, and N. Matni, "System level synthesis," *Annu. Rev. Control*, vol. 47, pp. 364–393, 2019. DOI: 10.1016/j.arcontrol.2019.03.006.

[52] 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.*, vol. 3, no. 1, pp. 1–122, 2010. DOI: 10.1561/2200000016.

[53] C. Amo Alonso, J. S. Li, J. Anderson, and N. Matni, "Distributed and localized model predictive control. Part I: Synthesis and implementation," *IEEE Transactions on Control of Network Systems*, pp. 1–12, 2022. DOI: 10.1109/TCNS.2022.3219770,

[54] D. Limon Marruedo, T. Alamo, and E. F. Camacho, "Stability analysis of systems with bounded additive uncertainties based on invariant sets: Stability and feasibility of MPC," in *Proc. IEEE ACC*, 2002, pp. 364–369. DOI: 10.1109/ACC.2002.1024831.

[55] J. Sieber, S. Bennani, and M. N. Zeilinger, "A system level approach to tube-based model predictive control," *IEEE Control Systems Letters*, vol. 6, pp. 776–781, 2022. DOI: 10.1109/LCSYS.2021.3086190.

[56] C. Amo Alonso and N. Matni, "Distributed and localized closed-loop model predictive control via System Level Synthesis," pp. 5598–5605, 2020. DOI: 10.1109/CDC42340.2020.9303936,

[57] Y. Chen and J. Anderson, "System level synthesis with state and input constraints," in *Proc. IEEE CDC*, 2019, pp. 5258–5263. DOI: 10.1109/CDC40024.2019.9029745.

[58] C. Amo Alonso, N. Matni, and J. Anderson, "Explicit distributed and localized model predictive control via System Level Synthesis," pp. 5606–5613, 2020. DOI: 10.1109/CDC42340.2020.9304349,

[59] J. S. Li, *SLS-MATLAB: Matlab toolbox for system level synthesis*, 2019. [Online]. Available: `https://github.com/sls-caltech/sls-code`.

[60] M. Grant and S. Boyd, *CVX: Matlab software for disciplined convex programming, version 2.1*, `http://cvxr.com/cvx`, 2014.

[61] C. Conte, T. Summers, M. N. Zeilinger, M. Morari, and C. N. Jones, "Computational aspects of distributed optimization in model predictive control," in *Proc. IEEE CDC*, IEEE, 2012, pp. 6819–6824. DOI: `10.1109/CDC.2012.6426138`.

[62] Q. Li, B. Kailkhura, R. Goldhahn, P. Ray, and P. K. Varshney, "Robust decentralized learning using admm with unreliable agents," *IEEE Transactions on Signal Processing*, vol. 70, pp. 2743–2757, 2022. DOI: `10.1109/TSP.2022.3178655`.

[63] S. Sadraddini and R. Tedrake, "Linear encodings for polytope containment problems," in *Proc. IEEE CDC*, 2019, pp. 4367–4372. DOI: `10.1109/CDC40024.2019.9029363`.

[64] C. Amo Alonso, J. S. Li, J. Anderson, and N. Matni, "Distributed and localized model predictive control. Part II: Theoretical guarantees," *IEEE Transactions on Control of Network Systems*, pp. 1–11, 2022. DOI: `10.1109/TCNS.2023.3262650`,

[65] F. Blanchini, "Set invariance in control," *Automatica*, vol. 35, no. 11, pp. 1747–1767, 1999. DOI: `10.1016/S0005-1098(99)00113-2`.

[66] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000. DOI: `10.1016/S0005-1098(99)00214-9`.

[67] J. Löfberg, "Minimax approaches to robust modelpredictive control," *PhD thesis, Department of ElectricalEngineering, Linköping University, Sweden*, 2003.

[68] S. P. Boyd and L. Vandenberghe, *Convex optimization*, en. Cambridge, UK; New York: Cambridge University Press, 2004, ISBN: 978-0-521-83378-3.

[69] K. Rokade and R. K. Kalaimani, "Distributed ADMM over directed networks," *arXiv preprint arXiv:2010.10421*, 2020. DOI: `10.48550/arXiv.2010.10421`.

[70] C. Amo Alonso, D. Ho, and J.M. Maestre, "Distributed linear quadratic regulator robust to communication dropouts," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 3072–3078, 2020, 21st IFAC World Congress. DOI: `10.1016/j.ifacol.2020.12.1012`,

[71] J. Yu, Y.-S. Wang, and J. Anderson, "Localized and distributed $\mathcal{H}_2$ state feedback control," pp. 2732–2738, 2021. DOI: `10.23919/ACC50511.2021.9483301`.

[72] E. Gilbert and K. Tan, “Linear systems with state and control constraints: The theory and applications of maximal output admissible sets,” *IEEE Trans. Autom. Control*, vol. 36, no. 9, pp. 1008–1020, 1991. DOI: 10.1109/9.83532.

[73] G. Costantini, R. Rostami, and D. Gorges, “Decomposition methods for distributed quadratic programming with application to distributed model predictive control,” in *IEEE Proc. Annu. Allerton Conf. Commun., Control, Comput.*, 2018, pp. 943–950. DOI: 10.1109/ALLERTON.2018.8636067.

[74] J. S. Li and C. Amo Alonso, “Global performance guarantees for localized model predictive control,” Submitted to *IEEE Open Journal of Control Systems*, 2023. DOI: 10.48550/arXiv.2303.11264,

[75] L. Ballotta, M. R. Jovanović, and L. Schenato, “Can decentralized control outperform centralized? the role of communication latency,” *IEEE Transactions on Control of Network Systems*, pp. 1–11, 2023. DOI: 10.1109/TCNS.2023.3237483.

[76] T. Summers and J. Ruths, “Performance bounds for optimal feedback control in networks,” in *Proc. IEEE ACC*, 2018, pp. 203–209. DOI: 10.23919/ACC.2018.8431774.

[77] J. S. Li, C. Amo Alonso, and J. C. Doyle, “Frontiers in scalable distributed control: SLS, MPC, and beyond,” in *IEEE American Control Conference*, 2021, pp. 2720–2725. DOI: 10.23919/ACC50511.2021.9483130.

[78] I. R. Shafarevich and A. O. Remizov, *Linear algebra and geometry*. Springer Science & Business Media, 2012. DOI: 10.1201/9781466593480.

[79] H. Y. Cheung, T. C. Kwok, and L. C. Lau, “Fast matrix rank algorithms and applications,” *Journal of the ACM (JACM)*, vol. 60, no. 5, pp. 1–25, 2013. DOI: 10.1145/2528404.

[80] C. Amo Alonso*, F. Yang*, and N. Matni, “Data-driven distributed and localized model predictive control,” *IEEE Open Journal of Control Systems*, vol. 1, pp. 29–40, 2022. DOI: 10.1109/OJCSYS.2022.3171787,

[81] I. Markovsky and F. Dörfler, “Behavioral systems theory in data-driven analysis, signal processing, and control,” *Annual Reviews in Control*, vol. 52, pp. 42–64, 2021. DOI: 10.1016/j.arcontrol.2021.09.005.

[82] F. Dörfler, P. Tesi, and C. De Persis, “On the certainty-equivalence approach to direct data-driven lqr design,” *IEEE Transactions on Automatic Control*, pp. 1–8, 2023. DOI: 10.1109/TAC.2023.3253787.

[83] S. Muntwiler, K. P. Wabersich, L. Hewing, and M. N. Zeilinger, “Data-driven distributed stochastic model predictive control with closed-loop chance constraint satisfaction,” pp. 210–215, 2021. DOI: 10.23919/ECC54610.2021.9655214.

[84] J. Willems and J. Polderman, *Introduction to Mathematical Systems Theory: A Behavioral Approach*, T. in Applied Mathematics, Ed. Springer, New York, 1997. DOI: 10.1007/978-1-4757-2953-5.

[85] L. Gurobi Optimization, *Gurobi optimizer reference manual*, 2021. [Online]. Available: http://www.gurobi.com.

[86] C. Amo Alonso and S.-H. Tseng, "Effective GPU parallelization of distributed and localized model predictive control," pp. 199–206, 2022. DOI: 10.1109/ICCA54724.2022.9831839,

[87] K. M. Abughalieh and S. G. Alawneh, "A survey of parallel implementations for model predictive control," en, *IEEE Access*, vol. 7, pp. 34 348–34 360, 2019. DOI: 10.1109/ACCESS.2019.2904240.

[88] H. Jonson and T. Glad, "A method for state and control constrained linear quadratic control problems," in *Proc. 9th IFAC World Cong., Budapest, Hungary,* 2-6 July, 1984, pp. 229–233. DOI: 10.1016/S1474-6670(17)61202-3.

[89] S. Richter, C. N. Jones, and M. Morari, "Real-time input-constrained MPC using fast gradient methods," in *Proc. IEEE CDC*, 2009, pp. 7387–7393. DOI: 10.1109/CDC.2009.5400619.

[90] F. Ke, Z. Li, H. Xiao, and X. Zhang, "Visual servoing of constrained mobile robots based on model predictive control," *IEEE Trans. on Syst., Man, and Cybern.: Syst.*, vol. 47, no. 7, pp. 1428–1438, 2017. DOI: 10.1109/TSMC.2016.2616486.

[91] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002. DOI: 10.1016/S0005-1098(01)00174-1.

[92] M. Kogel and R. Findeisen, "Parallel solution of model predictive control using the alternating direction multiplier method," in *Proc. 4th IFAC Conf. on Nonlinear Model Predictive Control*, 2012, pp. 369–374. DOI: 10.3182/20120823-5-NL-3013.00081.

[93] D.-K. Phung, B. Herisse, J. Marzat, and S. Bertrand, "Model predictive control for autonomous navigation using embedded graphics processing unit," in *Proc. 20th IFAC World Cong., Toulouse, France,* 9-14 July, 2017, pp. 11 883–11 888. DOI: 10.1016/j.ifacol.2017.08.1415.

[94] L. Yu, A. Goldsmith, and S. Di Cairano, "Efficient convex optimization on GPUs for embedded model predictive control," in *Proc. General Purpose GPUs, Austin, TX, USA*, 2017, pp. 12–21. DOI: 10.1145/3038228.3038234.

[95] P. Hyatt, C. S. Williams, and M. D. Killpack, *Parameterized and GPU-parallelized real-time model predictive control for high degree of freedom robots*, 2020. [Online]. Available: https://arxiv.org/abs/2001.04931.

[96] B. Plancher and S. Kuindersma, "Realtime model predictive control using parallel DDP on a GPU," in *Proc. IEEE ICRA*, Montreal, Canada, May 2019.

[97] K. Ling, B. Wu, and J. Maciejowski, "Embedded model predictive control (MPC) using a FPGA," in *17th IFAC World Cong., Seoul, Korea,* 6-11 July, 2008, pp. 15 250–15 255. DOI: 10.3182/20080706-5-KR-1001.02579.

[98] A. Alessio and A. Bemporad, "A Survey on Explicit Model Predictive Control," in *Nonlinear Model Predictive Control*, M. Morari, M. Thoma, L. Magni, D. M. Raimondo, and F. Allgöwer, Eds., vol. 384, Springer Berlin Heidelberg, 2009, pp. 345–369. DOI: 10.1007/978-3-642-01094-1_29.

[99] P. Grieder, M. Kvasnica, M. BaotiÄ, and M. Morari, "Stabilizing low complexity feedback control of constrained piecewise affine systems," *Automatica*, vol. 41, no. 10, pp. 1683–1694, Oct. 2005. DOI: 10.1016/j.automatica.2005.04.016.

[100] S. Hovland, J. T. Gravdahl, and K. E. Willcox, "Explicit model predictive control for large-scale systems via model reduction," *Journal of Guidance, Control, and Dynamics*, vol. 31, no. 4, pp. 918–926, Jul. 2008. DOI: 10.2514/1.33079.

[101] G. Pannocchia, J. B. Rawlings, and S. J. Wright, "Fast, large-scale model predictive control by partial enumeration," *Automatica*, vol. 43, no. 5, pp. 852–860, May 2007. DOI: 10.1016/j.automatica.2006.10.019.

[102] T. Geyer, F. D. Torrisi, and M. Morari, "Optimal complexity reduction of polyhedral piecewise affine systems," *Automatica*, vol. 44, no. 7, pp. 1728–1740, Jul. 2008. DOI: 10.1016/j.automatica.2007.11.027.

[103] J. Holaza, B. TakÃ¡cs, M. Kvasnica, and S. D. Cairano, "Nearly optimal simple explicit MPC controllers with stability and feasibility guarantees," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 667–684, 2015. DOI: 10.1002/oca.2131.

[104] S. Koehler and F. Borrelli, "Building temperature distributed control via explicit MPC and a Trim and Respond methods," in *2013 European Control Conference (ECC)*, Zurich: IEEE, Jul. 2013, pp. 4334–4339. DOI: 10.23919/ECC.2013.6669781.

[105] Y. Wang, N. Matni, and J. C. Doyle, "Separable and localized system-level synthesis for large-scale systems," *IEEE Trans. Autom. Control*, vol. 63, no. 12, pp. 4234–4249, 2018. DOI: 10.1109/TAC.2018.2819246.

[106] S.-H. Tseng and J. S. Li, *SLSpy: Python-based system-level controller synthesis framework*, 2020. [Online]. Available: `http://arxiv.org/abs/2004.12565`.

[107] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[108] J. S. Li, C. Amo Alonso, and J. C. Doyle, "Frontiers in scalable distributed control: SLS, MPC, and beyond," pp. 2720–2725, 2021. DOI: `10.23919/ACC50511.2021.9483130`,