Real-Time Optimization for Distributed Nonlinear Model Predictive Control

Vom Promotionsausschuss der
Technischen Universität Hamburg

zur Erlangung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)

genehmigte Dissertation (Monografie)

von
Gösta Stomberg

aus
Frankfurt am Main

2026

1. Gutachter: Prof. Dr.-Ing. Timm Faulwasser, Technische Universität Hamburg
2. Gutachter: Prof. Dr. Colin Jones, EPFL
3. Gutachter: Prof. Dr. Sébastien Gros, Norwegian University of Technology
Tag der mündlichen Prüfung: 22.04.2026

Abstract

The widespread availability of computing and communication technology promotes the development of Cyber Physical Systems of Systems (CPSoS) which couple subsystems in the physical and cyber domains. CPSoS enable cooperation between subsystems and thereby facilitate novel applications in diverse domains such as energy systems, robotic teams, vehicle fleets, and beyond. Distributed Model Predictive Control (DMPC) is a promising framework for control of CPSoS as it builds upon centralized Model Predictive Control (MPC)—a framework which is widely considered in academia *and* industry—while also addressing the needs of CPSoS. One promising avenue relies on the use of decentralized optimization algorithms to solve a cooperative Optimal Control Problem (OCP) online without a coordinating entity. However, the iterative nature of decentralized optimization induces the need for repeated communication between coupled subsystems, which may jeopardize real-time feasibility due to delays.

For centralized Nonlinear MPC (NMPC), real-time feasibility is achieved through Real-Time Iterations (RTIs) that enable fast control sampling and exhibit strong performance in implementations. This thesis presents a novel decentralized RTI (dRTI) scheme for DMPC. The proposed approach decomposes the computations of RTI schemes by exploiting the sparse coupling structure of CPSoS. The framework builds upon a novel decentralized Sequential Quadratic Programming (dSQP) method which parallelizes expensive computations among subsystems via the Alternating Direction Method of Multipliers (ADMM). We first prove the local convergence of dSQP to regular OCP solutions in the presence of non-convex constraints by viewing dSQP as an inexact Newton method. Subsequently, we show the local exponential stability of the closed-loop system-optimizer dynamics and we provide quantifiable bounds on the number of optimizer iterations that guarantee stability. This improves earlier results, because dRTI does not require centralized coordination, it does not rely on feasible initializations, and local optimizer convergence is rigorously shown and not assumed. From a computational point of view, dRTI requires subsystems to solve convex quadratic programs instead of nonlinear programs, which is beneficial in embedded implementations. We demonstrate the efficacy of dRTI in hardware experiments for the formation control of mobile robots and holonomic hovercraft, studying scenarios of increasing difficulty ranging up to collision avoidance in dynamic environments. There, we show that dRTI allows to run DMPC on embedded hardware at sampling frequencies of 20 Hz, which is five times faster than previously reported implementations. In addition to the above theoretical and practical contributions, we showcase the computational efficacy of dRTI through simulation case studies. Specifically, we consider linear-quadratic and nonlinear DMPC for frequency control problems in electric power grids with up to 1,800 states and 333,000 decision variables. For the considered scenarios, a parallelized dRTI implementation achieves nearly centralized closed-loop control performance but requires less than 7 % of the computation time needed by state-of-the-art interior point solvers.

Für meine Mutter, meinen Vater und meine Großeltern

Acknowledgements

This dissertation is the result of my studies as research associate, doctoral student, and visiting student at the Institute of Energy Systems, Energy Efficiency and Energy Economics at TU Dortmund University, the Institute of Control Systems at Hamburg University of Technology, and the Laboratoire d'Automatique at EPFL. This achievement would not have been possible without the extensive support I have received from colleagues, friends, and family members. First and foremost, I thank my advisor Prof. Dr.-Ing. Timm Faulwasser for his close mentorship and his relentless efforts to teach me about control, research, and beyond. Moreover, I thank Prof. Dr. Colin Jones and Prof. Dr. Sébastien Gros for their service as external reviewers of this dissertation. In particular, I thank Prof. Dr. Colin Jones for hosting me during a research visit to EPFL and I thank the NCCR Automation for supporting my visit to Lausanne. Special thanks go to Prof. Dr.-Ing. Timm Faulwasser, Prof. Dr.-Ing. Christian Rehtanz, Apl. Prof. Dr.-Ing. Ulf Häger, and the rest of ie^3 and ICS for providing pleasant and supportive environments in Dortmund and Hamburg. Notably, I thank Emily Elvermann, Tobias Loidl, Dirk Baack, Nicole Funke, Nina Ganser, and Kirsten Johanson for their support on technical and administrative matters. Furthermore, I thank Dr.-Ing. Alexander Engelmann for our productive collaboration, for sharing his unmatched fascination for numerical optimization, and for our countless discussions on the intricacies of ADMM. Moreover, I thank Prof. Dr.-Ing. Henrik Ebel and Prof. Dr.-Ing. Peter Eberhard for our joint work on the experimental validation of distributed MPC, which proved pivotal in shaping my doctoral studies towards real-time implementations. Likewise, I thank Prof. Dr. Moritz Diehl for our joint efforts on the stability analysis of decentralized real-time iterations, the key theoretical contribution of this dissertation. Special thanks go to Dr. Roland Schwan and Andrea Grillo for their unwavering commitment to our joint hovercraft experiments. Representative of all the students which I have had the pleasure to supervise, I especially thank Michael Kaupmann, Felix Greiwe, and Maurice Raetsch for their work on distributed MPC simulations. Additionally, I thank Felix Göke, Dr.-Ing. Fabian Kurtz, Patrick Lenzen, Fabian Menebröker, and Dr.-Ing. Sebastian Raczka for the stimulating joint student supervision. I further thank Maísa, Jens, Julian, Ruchuan, Guanru, and Oleksii for their companionship which enriched my time in Dortmund beyond teaching and research. Finally, I thank my grandparents, my father, and my mother for the faith they have placed in me throughout the years and for their unyielding support and commitment.

Content

Notation

Abbreviations and Acronyms

AC-OPF	Alternating Current Optimal Power Flow
ADMM	Alternating Direction Method of Multipliers
ALADIN	Augmented Lagrangian Alternating Direction Inexact Newton method
CG	Conjugate Gradients
CPSoS	Cyber Physical Systems of Systems
CPU	Central Processing Unit
DMPC	Distributed Model Predictive Control
DD	Dual Decomposition
DGP	Distributed Gradient Projection
dRTI	decentralized Real-Time Iterations
dSQP	decentralized Sequential Quadratic Programming
d-ASM	essentially decentralized Active Set Method
d-CG	essentially decentralized Conjugate Gradient method
d-IP	essentially decentralized Interior Point method
GPU	Graphics Processing Unit
KKT	Karush-Kuhn-Tucker
LCM	Lightweight Communications and Marshalling
LICQ	Linear Independence Constraint Qualification
LTI	Linear Time Invariant
MPC	Model Predictive Control
NMPC	Nonlinear Model Predictive Control
NLP	Nonlinear Program
PID	Proportional-Integral-Derivative
OCP	Optimal Control Problem
OPF	Optimal Power Flow
OTSA	Optimality Tracking Splitting Algorithm
QP	Quadratic Program
QCQP	Quadratically Constrained Quadratic Program
ROS	Robot Operating System
RTI	Real-Time Iterations
SB-DMPC	Sensitivity-Based Distributed Model Predictive Control
SOSC	Second-Order Sufficient Conditions
SQP	Sequential Quadratic Programming
TL-ALM	Two-Level Augmented Lagrangian Method

Mathematical Symbols

$\mathcal{A}(z^\star)$	index set of active inequality constraints at $z^\star$, $\mathcal{A}(z^\star) \subseteq \{1, \dots, n_h\}$
$\mathcal{D}$	index set of buses in an electrical grid, $\mathcal{D} \doteq \{1, \dots, \lvert\mathcal{D}\rvert\}$
E	centralized coupling matrix, $E \in \mathbb{R}^{n_c \times n}$
$\mathbb{E}$	feasible set of coupling constraints, $\mathbb{E} \subseteq \mathbb{R}^n$
f	centralized objective, $f : \mathbb{R}^n \to \mathbb{R}$
f^{c}	centralized continuous-time system dynamics, $f^{\mathrm{c}} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$
f^δ	centralized discrete-time system dynamics, $f^\delta : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$
$\tilde{f}$	frequency $\tilde{f} \doteq \omega/(2\pi)$, $\tilde{f} \in \mathbb{R}$
g	centralized equality constraints, $g : \mathbb{R}^n \to \mathbb{R}^{n_g}$
h	centralized inequality cosntraints, $h : \mathbb{R}^n \to \mathbb{R}^{n_h}$
F	residual, force
$\nabla f(x)$	gradient of a function $f : \mathbb{R}^n \to \mathbb{R}$, $\nabla f(x) \in \mathbb{R}^n$
$\nabla F(x)$	Transpose of the Jacobian matrix of a function $F = (F_1, \dots, F_m) : \mathbb{R}^n \to \mathbb{R}^m$, $\nabla F \doteq [\nabla F_1, \dots, \nabla F_m] \in \mathbb{R}^{n \times m}$
$\mathcal{G}$	OCP coupling and communication graph
H	Hessian matrix, $H \in \mathbb{R}^{n \times n}$
I	identity matrix of appropriate dimension
$\mathbb{I}$	set of integer numbers
$\mathcal{I}(z^\star)$	set of inactive inequality constraints at $z^\star$, $\mathcal{I}(z^\star) \doteq \{1, \dots, n_h\} \setminus \mathcal{A}(z^\star)$
ℓ	centralized stage cost, $\ell : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$
L	Lagrangian function $L : \mathbb{R}^n \times \mathbb{R}^{n_g} \times \mathbb{R}^{n_h} \times \mathbb{R}^{n_c} \to \mathbb{R}$
L_ρ	augmented Lagrangian function $L_\rho : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$
$\mathcal{L}$	index set of loads in an electrical grid, $\mathcal{L} \subseteq \mathcal{D}$
M_{avg}	ADMM averaging matrix, $M_{\mathrm{avg}} \in \mathbb{R}^{n \times n}$
N	OCP horizon, $N \in \mathbb{N}$
$\mathbb{N}$	set of natural numbers
$\mathbb{N}_0$	set of natural numbers including zero, $\mathbb{N}_0 \doteq \mathbb{N} \cup \{0\}$
$\mathcal{N}_i^{\mathrm{in}}$	index set of in-neighbors of subsystem $i \in \mathcal{S}$, $\mathcal{N}_i^{\mathrm{in}} \subseteq \mathcal{S}$
$\mathcal{N}_i^{\mathrm{out}}$	index set of out-neighbors of subsystem $i \in \mathcal{S}$, $\mathcal{N}_i^{\mathrm{out}} \subseteq \mathcal{S}$
$\mathcal{N}_i$	index set of neighbors of subsystem $i \in \mathcal{S}$, $\mathcal{N}_i \doteq \mathcal{N}_i^{\mathrm{in}} \cup \mathcal{N}_i^{\mathrm{out}} \subseteq \mathcal{S}$
p	primal-dual variables, $p \in \mathbb{R}^{n_p}$
$p^\star$	KKT point, $p^\star \in \mathbb{R}^{n_p}$
$(p_{\mathrm{x}}, p_{\mathrm{y}})$	robot position, $(p_{\mathrm{x}}, p_{\mathrm{y}}) \in \mathbb{R}^2$
P	state weight matrix for terminal penalty, $P \in \mathbb{R}^{n_x \times n_x}$ or active power, $P \in \mathbb{R}$
Q	state weight matrix for stage cost, $Q \in \mathbb{R}^{n_x \times n_x}$ or reactive power, $Q \in \mathbb{R}$

$\mathcal{P}$ index set of generators in an electrical grid, $\mathcal{P} \subseteq \mathcal{D}$
R input weight matrix for stage cost, $R \in \mathbb{R}^{n_u \times n_u}$
$\mathcal{S}$ index set of subsystems, $\mathcal{S} \doteq \{1, \ldots, S\}$
t discrete time index, $t \in \mathbb{N}_0$
t_c continuous time, $t_\mathrm{c} \in \mathbb{R}$
u centralized control input, $u \in \mathbb{R}^{n_u}$
$\boldsymbol{u}$ predicted centralized input trajectory, $u \in \mathbb{R}^{N \cdot n_u}$
$\mathbb{U}$ centralized input constraint set, $\mathbb{U} \subseteq \mathbb{R}^{n_u}$
V OCP value function, $V : \mathbb{R}^{n_x} \to \mathbb{R}$
V_f OCP terminal penalty, $V_\mathrm{f} : \mathbb{R}^{n_x} \to \mathbb{R}$
w copied states from in-neighbors, slack variable for d-IP
x centralized system state, $x \in \mathbb{R}^{n_x}$
$\boldsymbol{x}$ predicted centralized state trajectory, $x \in \mathbb{R}^{(N+1) \cdot n_x}$
$\mathbb{X}$ centralized state constraint set, $\mathbb{X} \subseteq \mathbb{R}^{n_x}$
z centralized decision variables, $z \in \mathbb{R}^n$
$z^\star$ local minimizer, $z^\star \in \mathbb{R}^n$
$\mathbb{Z}$ feasible set of the centralized subsystem constraints, $\mathbb{Z} \doteq \mathbb{R}^n$
$\underline{\mathbb{Z}}$ feasible set of a partially separable NLP, $\underline{\mathbb{Z}} \doteq \mathbb{Z} \cap \mathbb{E} \subseteq \mathbb{R}^n$
β terminal penalty scaling parameter, $\beta \geq 1$
γ consensus constraint Lagrange multiplier, $\gamma \in \mathbb{R}^n$
δ control sampling interval, d-IP barrier parameter, $\delta > 0$
ν centralized Lagrange multiplier to subsystem equality constraints, $\nu \in \mathbb{R}^{n_g}$
μ centralized Lagrange multiplier to subsystem inequality constraints, $\mu \in \mathbb{R}^{n_h}$
μ_c strong convexity parameter of a function, $\mu_\mathrm{c} > 0$
λ coupling constraint Lagrange multiplier, $\lambda \in \mathbb{R}^{n_c}$
ρ augmented Lagrangian penalty parameter, $\rho > 0$
φ angle of an inverted pendulum or orientation of a robot, $\varphi \in \mathbb{R}$
θ voltage angle of a bus in an electrical grid, $\theta \in \mathbb{R}$
ω angular velocity, $\omega \in \mathbb{R}$

Mathematical Notation

$\lvert \cdot \rvert$	cardinality of a set, magnitude of a scalar variable
$\lVert \cdot \rVert$	Chapters 1–3, 5–7, and Appendix A: any norm on $\mathbb{R}^n$ or its induced matrix norm. Chapter 4: Euclidean norm of a vector or spectral norm of a matrix.
$\lVert a \rVert_2$	Euclidean norm of a vector $a \in \mathbb{R}^n$, $\lVert a \rVert_2 \doteq \sqrt{a^\top a}$
$\sigma_{\max}(A)$	largest singular value of matrix A
$\lVert A \rVert_2$	Spectral norm of a matrix $A \in \mathbb{R}^{n \times m}$, $\lVert A \rVert \doteq \sigma_{\max}(A)$
$\lVert a \rVert_\infty$	maximum norm of a vector, $\lVert a \rVert_\infty \doteq \max(\lvert [a]_1 \rvert, \ldots, \lvert [a]_n \rvert)$
$\mathcal{B}(a, \varepsilon)$	open ε neighborhood of a point $a \in \mathbb{R}^n$, $\mathcal{B}(a, \varepsilon) \doteq \{b \in \mathbb{R}^n \mid \lVert b - a \rVert < \varepsilon\}$
$\bar{\mathcal{B}}(a, \varepsilon)$	closed ε neighborhood of a point $a \in \mathbb{R}^n$, $\bar{\mathcal{B}}(a, \varepsilon) \doteq \{b \in \mathbb{R}^n \mid \lVert b - a \rVert \leq \varepsilon\}$
$[a]_j$	j-th component of a vector a, $[a]_j \in \mathbb{R}$
$[A]_j$	j-th row of a matrix A, $[A]_j \in \mathbb{R}^{1 \times m}$
$[A]_{ij}$	component of a matrix at index (i, j), $[A]_{ij} \in \mathbb{R}$
$[a]_{\mathcal{A}}$	vector of components $[a]_j$ for all $j \in \mathcal{A}$
$[A]_{\mathcal{A}}$	matrix of rows $[A]_j$ for all $j \in \mathcal{A}$
$[A_{ij}]_{i,j \in \mathcal{S}}$	block matrix with block A_{ij} at block position (i, j)
$\mathbb{I}_{[0,N]}$	integer numbers from 0 to N, $\mathbb{I}_{[0,N]} = \{0, \ldots, N\}$
$A = \operatorname{diag}(a_1, \ldots, a_n)$	diagonal matrix with components $[A]_{ii} = a_i$, $A \in \mathbb{R}^{n \times n}$
$A = \operatorname{diag}(A_1, \ldots, A_S)$	block diagonal matrix with entry A_i at block position (i, i)
(x, y)	vertical concatenation of column vectors x and y into one column vector
$(x_i)_{i \in \mathcal{S}}$	concatenation $x \doteq (x_1, \ldots, x_S)$
$[A, B]$	horizontal concatentation of matrices $A \in \mathbb{R}^{n \times m_1}$ and $B \in \mathbb{R}^{n \times m_2}$, $[A, B] \in \mathbb{R}^{n \times (m_1 + m_2)}$
$\min(x, y)$	vector of component-wise minimum values of vectors x and y, $\min(x, y) \doteq (\min([x]_1, [y]_1), \ldots, \min([x]_n, [y]_n))$
$\mathbb{1}$	vector with all components equal to 1, $\mathbb{1} \in \mathbb{R}^{n_h}$
$\cdot_i$	quantity associated with subsystem $i \in \mathcal{S}$
$\cdot^k$	iteration of an optimization algorithm
$\cdot^{k,l}$	l-th inner iteration of the k-th outer iteration of a bi-level algorithm
$\cdot[\tau]$	predicted variable for stage $\tau \in \mathbb{I}_{[0,N]}$
$A^\top$	transpose of a matrix $A \in \mathbb{R}^{n \times m}$, $A^\top \in \mathbb{R}^{m \times n}$
$A \oplus B$	Minkowski sum of two sets A and B, $A \oplus B \doteq \{a + b \mid a \in A, b \in B\}$

1 Introduction

The continuous progress in computing gives rise to an ever-increasing number of *Cyber-Physical Systems* which combine communication, control, and computing with physical systems [Rajkumar et al. 2010; Kim and Kumar 2012]. Modern communication networks are fast and reliable and they foster the realization of ensembles of Cyber-Physical Systems, so-called *Cyber-Physical Systems of Systems* (CPSoS). These systems enable cooperation between subsystems and thus provide additional functionality and greater performance compared to individual Cyber-Physical Systems [Allgöwer et al. 2019]. In many application domains, CPSoS are sometimes referred to as *smart*, e.g. *smart* transportation, *smart* manufacturing, and *smart* grids. The latter illustrate the relevance of CPSoS particularly well, because energy systems are critical infrastructure for any modern society and they exhibit many of the characteristic properties of CPSoS: They are large in scale, interconnected in the physical and cyber domains, and spatially distributed. While power grids already incorporate many elements of CPSoS, the transition towards sustainability requires to adopt even further technologies attributed to CPSoS. Specifically, the integration of renewable distributed energy resources needs active—or *smart*— distribution grids as transmission system operators have little observation and lack control over these new installations [Allgöwer et al. 2019]. Thus, existing centralized control architectures for the operation of power grids will possibly be combined with distributed methods where subsystems cooperate to meet application demands [Molzahn et al. 2017].

CPSoS, distributed control, and the control of critical infrastructure networks constitute three key contemparory research challenges due to the paramount need for scalable methods which foster cooperation among subsystems [Lamnabhi-Lagarrigue et al. 2017]. Computational complexity, reliability, and limited communication bandwidth form crucial bottlenecks for centralized control of CPSoS, where a single controller governs all subsystems [Scattolini 2009], cf. Figure 1.1. Instead, distributed control assigns one controller to each subsystem and communication between the controllers of coupled subsystems improves performance. Crucially, distributed control does not require a centralized coordinator, which avoids a single point of failure and thus promotes resilience.

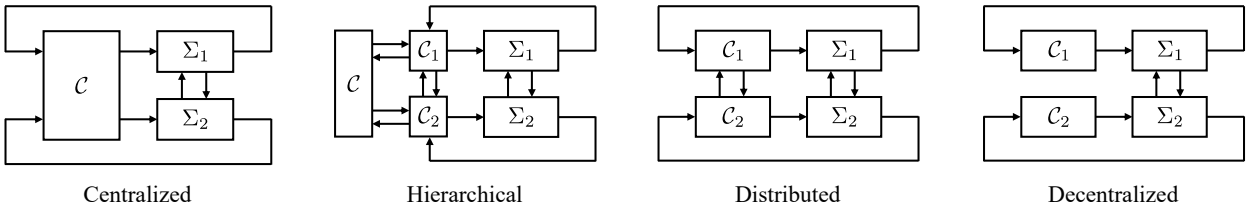


Figure 1.1: Control architectures for CPSoS as proposed in [Scattolini 2009]. Diagrams inspired by [Scattolini 2009] and figure adapted from [Stomberg et al. 2023].

As mentioned above, research on the distributed control of CPSoS is driven by different applications such as energy systems, water distribution networks, and traffic management [El Fawal et al. 1998; Kersbergen et al. 2016; Molzahn et al. 2017; Havlena et al. 2014; Li and De Schutter 2022]. Further target domains are multi-robot systems, spacecraft formations, automotive control, and building control [Bullo et al. 2009; Ramirez-Riberos et al. 2010; Guanetti et al. 2018; Lefebure et al. 2022]. For these applications, centralized control paradigms can be translated to distributed control by adapting the methods to the graph structures present in CPSoS and we refer to the textbooks [Lunze 2022; Bullo 2024] for recent treatments on distributed output and state feedback control. An alternative branch of research discusses tailored variants of Model Predictive Control (MPC) for CPSoS. The latter can be traced back at least to [Mesarovic et al. 1970] and forms the main topic of this thesis.

1.1 Model Predictive Control Architectures

We begin our introduction to predictive control with centralized Nonlinear MPC (NMPC) based on nonlinear discrete-time systems of the form

$$x(t+1) = f^{\delta}(x(t), u(t)), \quad x(0) = x_0, \tag{1.1}$$

where $\delta > 0$ is the control sampling interval, $x \in \mathbb{X} \subseteq \mathbb{R}^{n_x}$ is the system state, $u \in \mathbb{U} \subseteq \mathbb{R}^{n_u}$ is the control input, $f^{\delta} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ are the system dynamics, $t \in \mathbb{N}_0$ is the discrete time index, $\mathbb{N}_0$ are the natural numbers extended by zero, and $x_0 \in \mathbb{R}^{n_x}$ is the initial state. The MPC control input $u(t)$ is determined by solving a discrete-time Optimal Control Problem (OCP) such as

$$\min_{\boldsymbol{x},\boldsymbol{u}} \sum_{\tau=0}^{N-1} \ell(x[\tau], u[\tau]) + \beta V_{\mathrm{f}}(x[N]) \tag{1.2a}$$

subject to

$$x[\tau+1] = f^{\delta}(x[\tau], u[\tau]) \quad \forall \tau \in \mathbb{I}_{[0,N-1]}, \tag{1.2b}$$

$$x[0] = x(t), \tag{1.2c}$$

$$x[\tau] \in \mathbb{X} \quad \forall \tau \in \mathbb{I}_{[0,N]}, \tag{1.2d}$$

$$u[\tau] \in \mathbb{U} \quad \forall \tau \in \mathbb{I}_{[0,N-1]}. \tag{1.2e}$$

Let $(a, b) \in \mathbb{R}^{n_a+n_b}$ denote the concatenation of two vectors $a \in \mathbb{R}^{n_a}$ and $b \in \mathbb{R}^{n_b}$ into one column vector and let $\mathbb{I}_{[0,N]}$ denote the integers in the range $[0, N]$. The decision variables in OCP (1.2) are the predicted state and input trajectories $\boldsymbol{x} \doteq (x[0], \ldots, x[N]) \in \mathbb{R}^{(N+1)\cdot n_x}$ and $\boldsymbol{u} \doteq (u[0], \ldots, u[N-1]) \in \mathbb{R}^{N\cdot n_u}$, where $N \in \mathbb{N}$ is the prediction horizon, $\mathbb{N}$ are the natural numbers, and predicted variables are denoted by square brackets to distinguish between open-loop and closed-loop trajectories. The stage cost $\ell : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ and terminal penalty $V_{\mathrm{f}} : \mathbb{R}^n \to \mathbb{R}$

encode the control objective, e.g. by penalizing deviations from a setpoint. OCP (1.2) includes a scaling parameter $\beta \geq 1$ for the terminal penalty to avoid additional constraints on the terminal state $x[N]$ [Limon et al. 2006] . The control input at time t is then selected as the first part of the optimal input trajectory $\boldsymbol{u}^\star$, i.e. $u(t) \doteq u^\star[0]$. MPC has been widely adopted in industrial applications [Qin and Badgwell 2003] and we refer to [Maciejowski 2002; Camacho and Bordons 2007; Grüne and Pannek 2017; Rawlings et al. 2019] for more details on centralized MPC.

Predictive control is a prime candidate for controlling CPSoS due to its versatility with respect to the system model used for prediction, dimensions of control input and system output, and constraints on inputs, states, and functions thereof. The intrinsic applicability to multi-input-multi-output systems facilitates the simple conceptualization of centralized MPC schemes as described above [Scattolini 2009]. Essentially, this approach views all subsystems as a single dynamical system of large dimension which is then controlled using MPC. However, centralized MPC may be prohibitive due to application-specific considerations such as the lare geographic scale of power systems [Venkat et al. 2008] or because of the computational complexity associated with solving large-scale OCPs online. Thus, tailored MPC variants have been proposed for CPSoS which can be grouped into the control architectures of Figure 1.1 [Scattolini 2009]. For instance, decentralized MPC alleviates the computational bottleneck of centralized MPC by assigning one controller to each subsystem [Cohen 1977]. These controllers do not communicate with each other, which reduces computational complexity but is often insufficient for strongly coupled subsystems [Scattolini 2009]. Moreover, decentralized control fails to capitalize on key CPSoS technologies such as modern communication networks, views other subsystems as distubances, and thus compromises cooperation and ultimately performance. To balance the coordination and performance of centralized MPC with the resilience and scalability of decentralized MPC, hierarchical and Distributed MPC (DMPC) schemes have been proposed [Scattolini 2009; Christofides et al. 2013]. Hierarchical control schemes comprise a centralized coordinator as well as individual controllers for the subsystems and, in the context of MPC, date back at least to the 1970s [Mesarovic et al. 1970]. The computational restrictions of centralized MPC are alleviated by shifting costly computations to the subsystems, but the drawbacks of having a single point of failure persist.

To overcome this issue, DMPC designs should alleviate the need for a coordinator while requiring only neighbor-to-neighbor communication. Communicated variables between controllers typically entail predicted trajectories such that each subsystem can adapt its behavior based on the forecasts of neighbors. A common taxonomy in DMPC is to distinguish between *noniterative* and *iterative* schemes, i.e., methods with one or many communication rounds per control step [Scattolini 2009]. One may also distinguish between *non-cooperative* and *cooperative* methods, i.e., schemes where the subsystems pursue individual versus joint objectives. Notably, different definitions exist on possible combinations between iterative and cooperative dimensions. For instance, [Müller and Allgöwer 2017] allows for cooperative sequential schemes, whereas [Scattolini 2009] only allows iterative schemes to be called cooperative, claiming mutual consideration between subsystems can only result from iterative optimization. Throughout this thesis we fol-

low the latter notion and extend the definition of [Scattolini 2009] to the case of early optimizer termination, even if this results in only one communication round per time step.

1.2 Cooperative Distributed Model Predictive Control – State of the Art

This section summarizes the state of the art in cooperative DMPC and highlights the limitations of existing approaches. While the focus is primarily on cooperative DMPC, we occasionally give reference to other approaches such as sequential DMPC or hierarchical MPC. For detailed surveys on DMPC beyond cooperative schemes, we refer to [Scattolini 2009; Christofides et al. 2013; Müller and Allgöwer 2017].

1.2.1 Problem Statement

We assume that the centralized system (1.1) is partitioned into an index set $\mathcal{S} = \{1, \dots, S\}$ of subsystems, each of which is governed by the system dynamics

$$x_i(t+1) = f_i^\delta(x_i(t), u_i(t), x_{\mathcal{N}_i^{\text{in}}}(t)) \quad x_i(0) = x_{i,0}.$$

For all subsystems $i \in \mathcal{S}$, $x_i \in \mathbb{X}_i \subseteq \mathbb{R}^{n_{x_i}}$ is the subsystem state, $u_i \in \mathbb{U}_i \subseteq \mathbb{R}^{n_{u_i}}$ is the input, $f_i^\delta \to \mathbb{R}^{n_{x_i}} \times \mathbb{R}^{n_{u_i}} \times \mathbb{R}^{n_i^{\text{in}}} \to \mathbb{R}^{n_{x_i}}$, and $x_{i,0} \in \mathbb{R}^{n_{x_i}}$. Given an index set $\mathcal{N}$, let $(a_j)_{j\in\mathcal{N}}$ denote the vertical concatenation of the vectors a_j for all $j \in \mathcal{N}$ into one column vector. The vector $x_{\mathcal{N}_i^{\text{in}}} \doteq (x_j)_{j\in\mathcal{N}_i^{\text{in}}} \in \mathbb{R}^{n_i^{\text{in}}}$ collects the state vectors of all *in-neighbors* in the index set $\mathcal{N}_i^{\text{in}} \subseteq \mathcal{S}$ which we will define below. At time t, the subsystems cooperatively solve the OCP

$$\min_{\boldsymbol{x},\boldsymbol{u}} \sum_{i\in\mathcal{S}} \left(\sum_{\tau=0}^{N-1} \ell_i\left(x_i[\tau], u_i[\tau], x_{\mathcal{N}_i^{\text{in}}}[\tau]\right) + \beta V_{\text{f}_i}(x_i[N]) \right) \tag{1.3a}$$

subject to for all $i \in \mathcal{S}$

$$x_i[\tau+1] = f_i^\delta\left(x_i[\tau], u_i[\tau], x_{\mathcal{N}_i^{\text{in}}}[\tau]\right) \quad \forall \tau \in \mathbb{I}_{[0,N-1]}, \tag{1.3b}$$

$$x_i[0] = x_i(t), \tag{1.3c}$$

$$x_i[\tau] \in \mathbb{X}_i \qquad \forall \tau \in \mathbb{I}_{[0,N]}, \tag{1.3d}$$

$$u_i[\tau] \in \mathbb{U}_i \qquad \forall \tau \in \mathbb{I}_{[0,N-1]}, \tag{1.3e}$$

$$(x_i[\tau], x_j[\tau]) \in \mathbb{X}_{ij} \qquad \forall j \in \mathcal{N}_i^{\text{in}}, \quad \forall \tau \in \mathbb{I}_{[0,N]}. \tag{1.3f}$$

Each subsystem then selects its control input from the OCP solution, i.e. $u_i(t) \doteq \boldsymbol{u}_i^\star[0]$ for all $i \in \mathcal{S}$. As shown in Figure 1.2, the *cooperative OCP* (1.3) is a decomposed version of the centralized OCP (1.2) which arises from the partition of the centralized system into subsystems, see [Chanfreut et al. 2021b] for partitioning methods in the context of DMPC. That is, OCPs (1.2) and (1.3)

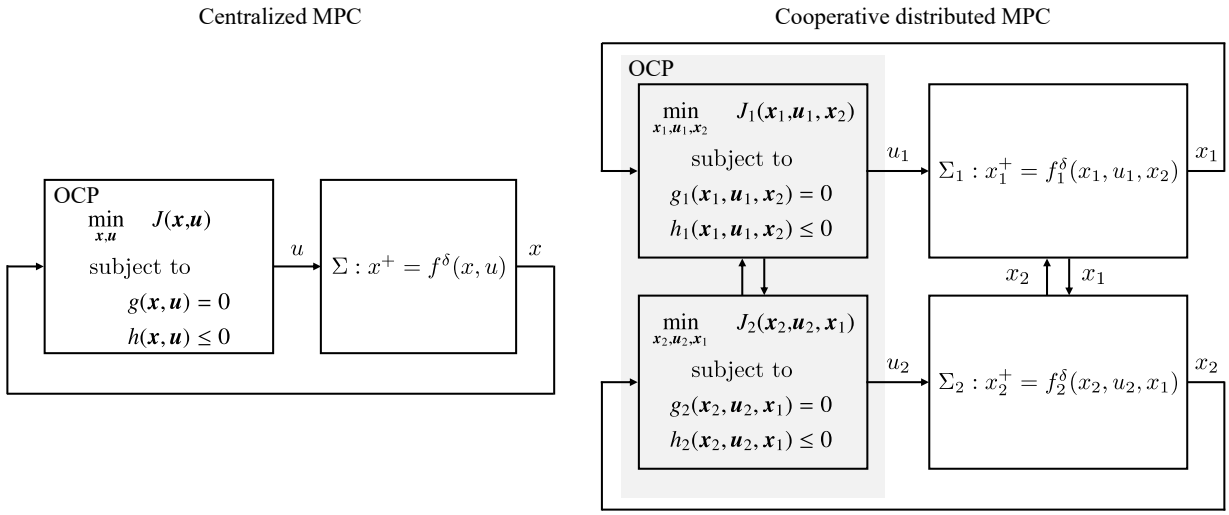


Figure 1.2: Partitioning of a centralized MPC scheme into subsystems for cooperative DMPC. The cooperative OCP shaded in gray is decomposed among individual controllers and is equivalent to the centralized OCP on the left. A distributed control scheme arises because the cooperative OCP is solved with decentralized optimization algorithms, where the exchanged variables between controllers depend on the type of optimization algorithm. The functions J, g, and h represent the objective, equality constraints, and inequality constraints in abstract form and the notation x^+ is a shorthand for $x(t+1)$.

are equivalent from a mathematical point of view. The centralized components read $x = (x_i)_{i\in\mathcal{S}}$, $u = (u_i)_{i\in\mathcal{S}}$,

$$\mathbb{X} \doteq \left\{ x \in \mathbb{R}^{n_x} \mid x_i \in \mathbb{X}_i,\ (x_i, x_j) \in \mathbb{X}_{ij}\ \forall j \in \mathcal{N}_i^{\text{in}},\ \forall i \in \mathcal{S} \right\}, \quad \mathbb{U} \doteq \mathbb{U}_1 \times \cdots \times \mathbb{U}_S,$$

and similarly for the centralized system dynamics f^δ and the OCP objective.

Since the centralized and cooperative OCPs are equivalent in theory, the key essence of cooperative DMPC is to solve the OCP without centralized computation in order to obtain a distributed control scheme. This can be done using iterative *decentralized optimization* algorithms which exploit the sparse coupling between subsystems. The coupling in OCP (1.3) can be described by a graph $\mathcal{G}(\mathcal{S}, \mathcal{E})$ with edges $\mathcal{E} \subseteq \mathcal{S} \times \mathcal{S}$, where the coupling of subsystem $i \in \mathcal{S}$ is captured by the sets of *in-* and *out-neighbors*

$$\mathcal{N}_i^{\text{in}} \doteq \{j \in \mathcal{S} \setminus \{i\} \mid (j,i) \in \mathcal{E}\} \quad \text{and} \quad \mathcal{N}_i^{\text{out}} \doteq \{j \in \mathcal{S} \setminus \{i\} \mid (i,j) \in \mathcal{E}\}. \tag{1.4}$$

The set $\mathcal{N}_i^{\text{in}}$ collects all subsystems which directly influence subsystem i and vice versa for $\mathcal{N}_i^{\text{out}}$. Note that this neighborhood definition allows not only for coupling in the system dynamics, but also in the objectives and constraints. For instance, neighbors could be other buses in a power grid because of coupled dynamics or other robots in a multi-robot system because of a joint control objective or collision avoidance constraints.

Assumption 1.1 (Bi-directional communication)**.** *Throughout the thesis, we assume that all subsystems $i \in \mathcal{S}$ can communicate with their neighbors $\mathcal{N}_i \doteq \mathcal{N}_i^{\text{in}} \cup \mathcal{N}_i^{\text{out}}$ to solve OCP* (1.3) *online.* □

Remark 1.1 (Connected coupling graph $\mathcal{G}$)**.** *Without loss of generality, we assume the undirected coupling graph $\mathcal{G}$ to be connected. If $\mathcal{G}$ is disconnected, then the centralized system can be trivially split into independent CPSoS, each of which can be controlled individually.* □

Origin and advantages of cooperative DMPC Cooperative DMPC schemes as outlined above date back at least to [El Fawal et al. 1998; Du et al. 2001; Jia and Krogh 2001]. Rooted in hierarchical MPC and *decomposition-coordination* optimization methods [Mesarovic et al. 1970; Cohen 1978], the pioneering scheme in [El Fawal et al. 1998] for water irrigation networks exhibits two charateristic design elements still commonly used in state-of-the-art cooperative DMPC: First, subsystems share predictions with neighbors through copied state variables. Second, the OCP is solved via an augmented Lagrangian-based optimization algorithm to decompose the coupling between original and copied variables. The main advantage of cooperative DMPC is that, in principle, the strong performance of centralized MPC can be attained as OCPs (1.2) and (1.3) are equivalent. However, inevitable computation and communication delays in decentralized optimization challenge the real-time feasibility and limit the number of optimizer iterations which can be performed in each control step. Thus, a crucial factor in the design and implementation of cooperative DMPC is the choice of the decentralized optimization method which solves OCP (1.3) online. We defer a detailed discussion of possible algorithms to Chapter 2 and instead first focus on the design, experimental validation, and numerical performance of cooperative DMPC.

1.2.2 OCP Design and Stability

Since the cooperative OCP (1.3) is a reformulation of the centralized OCP (1.2), the design and stability analysis of cooperative DMPC typically adapts prior results from centralized NMPC to the sparse coupling structure of CPSoS.

Terminal ingredients A common approach for designing stabilizing (N)MPC schemes is to incorporate a tailored terminal penalty V_f and to enforce an additional constraint on the terminal state $x[N]$ [Mayne et al. 2000]. If the linearized system (A, B) formed at the setpoint is stabilizable and if $\ell(x, u) \doteq x^\top Q x/2 + u^\top R u/2$ where $Q \in \mathbb{R}^{n_x \times n_x}$ is positive semi-definite such that (A, Q) is detectable and where $R \in \mathbb{R}^{n_u \times n_u}$ is positive definite, then a suitable terminal penalty can be found by solving the algebraic Riccati equation [Rawlings et al. 2019]. However, this approach is not directly applicable to DMPC, because the resulting weight matrix $P \in \mathbb{R}^{n_x \times n_x}$ is generally dense. This challenge is addressed for linear systems with coupling in the dynamics through the tailored design of separable terminal penalties and constraints in [Conte et al. 2016]. Similar approaches that further allow for coupled state constraints are presented in [Darivianakis et al. 2020; Aboudonia et al. 2020] while [Wang and Manzie 2022] further presents a robust design which additionally

considers bounded disturbances in the system dynamics. Moreover, the terminal ingredients can be adapted online to allow for piecewise-constant reference signals [Aboudonia et al. 2022a].

Modeling, partitioning, and plug-and-play Most DMPC approaches assume that state space models are available. Alternatives are to model uncertain coupled nonlinear dynamics as Gaussian processes [Grancharova et al. 2023] or, in the case of Linear Time-Invariant (LTI) systems, to employ data-driven predictive control based on behavioral systems theory [Allibhoy and Cortés 2021]. Moreover, OCP (1.3) assumes that the coupling graph $\mathcal{G}$ is given and time invariant. Depending on the application, $\mathcal{G}$ can also be designed in order to balance performance and coordination, accelerate decentralized optimization, and increase resilience with respect to failed communication links [Chanfreut et al. 2021b]. A related problem concerns the dynamic reconfiguration of DMPC if subsystems enter or leave the network. This is addressed in *plug-and-play* DMPC schemes by updating the terminal ingredients [Zeilinger et al. 2013; Aboudonia et al. 2022b] or by imposing additional constraints on coupled states [Lucia et al. 2015]. For multi-robot systems where communication links require proximity between neighbors, the connectivity of $\mathcal{G}$ can be enforced in the OCP [Carron et al. 2023].

Closed-loop stability The real-time requirements in control only allow for a finite number of optimizer iterations per control step such that any cooperative DMPC implementation must cope with suboptimal control inputs. Early approaches to guarantee stability of suboptimal linear and nonlinear DMPC employ feasible-side convergent algorithms [Stewart et al. 2010; Stewart et al. 2011]. Stability is then inferred using standard arguments for the stability of centralized NMPC [Mayne et al. 2000]. However, these DMPC algorithms either require all subsystems to access the centralized system dynamics or need feasible initializations. The former might violate privacy concerns of the subsystems whereas the latter is in general difficult to obtain without centralized coordination. For linear-quadratic DMPC, i.e., DMPC schemes where the OCP is a convex Quadratic Program (QP), these limitations can be overcome by Dual Decomposition (DD) at the cost of infeasible optimizer iterates. Stability despite the primal infeasibility of DD can be ensured via constraint tightening in combination with a sufficient OCP horizon [Giselsson and Rantzer 2014] or terminal ingredients [Köhler et al. 2019]. For nonlinear DMPC based on the Alternating Direction Method of Multipliers (ADMM), a stopping criterion for terminating ADMM in each control step while guaranteeing stability is derived in [Bestler and Graichen 2019]. However, therein ADMM is assumed to converge linearly to the OCP minimizer, which is a strong assumption as convergence guarantees which cover the decentralized ADMM implementations common in DMPC *and* non-convex constraints appear to be yet unavailable. Moreover, stopping criteria complicate real-time implementations and often introduce global communication requirements among all subsystems. As an alternative to ADMM, a decentralized augmented Lagrangian method for real-time DMPC is presented in [Hours and Jones 2016]. The bi-level optimization method enjoys linear convergence guarantees for sufficiently many inner iterations per outer iter-

ation. Moreover, closed-loop convergence of the optimizer is derived under the assumption that the changes in the system state between subsequent control samples are small, e.g. for fast sampling frequencies. However, the convergence result only refers to the optimizer and stability of the system is not considered. In contrast, stability guarantees which first prove the linear convergence of the optimal control algorithm and then infer stability for continuous-time nonlinear DMPC are available for Sensitivity-Based DMPC (SB-DMPC) [Pierer von Esch et al. 2025b]. SB-DMPC has appeared shortly after the framework proposed in this thesis and we provide more details on SB-DMPC in Chapter 4.

Thus, there appear to be no prior stability results for cooperative nonlinear DMPC which encompass both system *and* optimizer, allow for suboptimal inputs as well as infeasible optimizer iterates, and do not require online stopping criteria or a centralized coordinator.

1.2.3 Experimental Validation

Compared to the manifold efforts on the theoretical foundations of DMPC, the validation in hardware experiments is rare. Therefore, this section not only surveys results for cooperative DMPC, but also for non-cooperative and hierarchical schemes.

Non-cooperative DMPC and hierarchical MPC Experiments with non-cooperative DMPC require a great level of domain knowledge to achieve desired closed-loop properties with only few communication rounds per control step. For robot formation control, early results in this direction require the communication of predicted trajectories between neighbors and let each robot solve an individual OCP once per control step [Richards and How 2005]. Similar approaches are tested with centralized computation, i.e., with one computer performing the computations for all subsystems, on a four tank system [Mercangöz and Doyle 2007], a tractor-trailer combination [Kayacan et al. 2014], mobile robots [Mehrez et al. 2017], and quadcopters [Luis and Schoellig 2019; Luis et al. 2020]. For distributed computation, i.e., for experiments where neighboring subsystems in the coupling graph run on distinct computing hardware, related schemes have been applied to mobile robots [Kuwata et al. 2007; Kanjanawanishkul and Zell 2008a; Kanjanawanishkul and Zell 2008b], vehicles [Katriniok et al. 2022], and quadcopters [Vargas et al. 2022; Erunsal et al. 2024]. A hierachical MPC scheme based on a distributed SQP method is applied to the coordination of vehicles at intersections in [Hult et al. 2020]. The SQP scheme requires subsystems to solve small-scale convex subproblems and performs a line search on a central coordinator. The line search improves the convergence radius of the SQP scheme at the cost of centralized computation due to an ℓ_1 merit function. The experiments feature distributed computation and use a tailored wireless protocol for vehicle-to-vehicle communication.

Cooperative DMPC Motivated by the different needs in applications, the above experiments on non-cooperative DMPC employ a wide range of tailored algorithms. As summarized in Fig-

ure 1.3, the validation of cooperative DMPC instead appears to focus on only four decentralized optimization algorithms: Jacobi iterations [Stewart et al. 2010], Distributed Gradient Projection (DGP) [Stewart et al. 2011], ADMM [Kögel and Findeisen 2012; Summers and Lygeros 2012], and SB-DMPC [Pierer von Esch et al. 2025b]. Linear-quadratic DMPC based on Jacobi iterations with centralized computation is tested on a four tank system in [Ferramosca et al. 2013] and on mobile robots in [Ebel and Eberhard 2021]. While Jacobi iterations are feasible-side convergent, the DMPC scheme requires subsystems to incorporate system models from neighbors and needs a feasible initialization which complicates distributed implementations. In a similar vein, the nonlinear DMPC scheme based on DGP guarantees feasibility of the optimizer iterates, but requires all-to-all communication between subsystems [Stewart et al. 2011]. This approach has been tested in mobile robots experiments with centralized computation [Rosenfelder et al. 2022] and distributed computation [Ebel et al. 2024]. DMPC based on ADMM overcomes these limitations, but achieves feasibility only asymptotically. The experimental validation of DMPC using ADMM dates back at least to [Hentzelt et al. 2014], where a nonlinear DMPC scheme is devised for a four tank system. The experiments feature distributed and embedded execution by assigning one Raspberry Pi to each subsystem for running ADMM. The Raspberry Pis communicate with neighbors via Ethernet and the nonlinear subproblems in ADMM are solved with a tailored real-time gradient method [Graichen and Käpernick 2012], performing 15 ADMM iterations per NMPC step with approximately 200 ms computation time. A similar nonlinear DMPC scheme is applied to the formation control of mobile robots in dynamic environments in [Van Parys and Pipeleers 2017]. Therein, the robot kinematics are represented via splines, which allows to enforce collision avoidance constraints continuously over the prediction horizon. However, the large computational footprint of solving the subsystem Nonlinear Programs (NLPs) onboard with ipopt only allows to run one ADMM iteration within each 250 ms sampling interval [Wächter and Biegler 2006]. Faster control sampling with more ADMM iterations is considered in [Burk et al. 2021a] which runs a distributed ADMM implementation on a single PC.

It deserves to be remarked that all of the above experiments with sampling intervals in the Millisecond range feature systems with decoupled dynamics. A notable difference thus is the experimental validation of SB-DMPC on physically coupled magnetic levitation devices in [Pierer von Esch et al. 2025a]. There, nonlinear DMPC runs without centralized computation on Raspberry Pis which communicate via Ethernet, achieving sampling intervals of 200 ms. As for the theoretical contribution [Pierer von Esch et al. 2025b], the experiments in [Pierer von Esch et al. 2025a] appeared after the results to be presented in this work and we provide more details in Chapter 5.

To summarize, experimental results for cooperative DMPC based on ADMM have demonstrated satisfactory performance with few optimizer iterations and do not require feasible initializations. However, solving NLPs online can be a bottleneck for embedded implementations, convergence is not guaranteed for non-convex constraints, and prior distributed implementations only accommodate control sampling frequencies of up to 4 Hz.

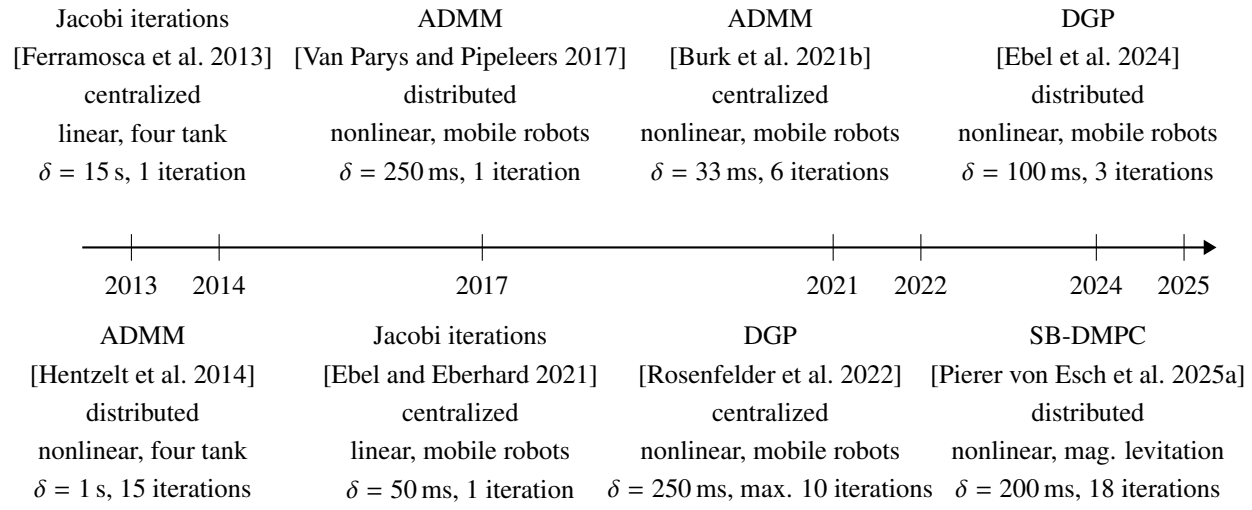


Figure 1.3: Timeline of cooperative DMPC hardware experiments with employed optimization algorithm, reference, type of computation, class of prediction model, system to be controlled, control sampling interval δ, and number of optimizer iterations per control step. The computation architecture is called distributed, if neighboring subsystems compute on distinct machines. In centralized computation, the computations of all or some neighboring subsystems run on the same machine.

1.2.4 Computational Performance

A key reason for the development of DMPC is to remedy the computational limits of centralized control for large-scale CPSoS. Unless centralized schemes prove impractical due to application-specific considerations such as privacy concerns, the computational performance of DMPC for large networks is pivotal to justify the engineering overhead associated with the design, analysis, and real-time implementation of cooperative DMPC compared to centralized MPC.

Centralized algorithms for large-scale optimization Decomposing large-scale OCPs to improve computational tractability is not a unique feature of DMPC as can be seen from the rich literature on *parallel MPC* [Abughalieh and Alawneh 2019; Bernal Neira et al. 2024]. For instance, parallel linear algebra packages offer an easy-to-use option to improve the scalability of centralized optimizers such as the Operator Splitting Quadratic Program (OSQP) solver [Stellato et al. 2020; Intel 2024]. Alternatively, *internal decomposition* methods can exploit problem-level sparsity, e.g. for accelerating the solution of the Karush-Kuhn-Tucker (KKT) system inside interior points methods [Zavala et al. 2008]. More recent efforts have focused on the deployment of centralized solvers on Graphics Processing Units (GPUs) to exploit their massive parallelism. A parallel GPU implementation of ADMM for convex QPs is presented in [Yu et al. 2017]. There, the computationally expensive step of solving an equality-constrained QP is done via direct solution of the KKT system and the corresponding matrix-vector multiplication is parallelized on a GPU. For OSQP, the costly

solution of an equality-constrained QP within each optimizer iteration is parallelized on a GPU via a preconditioned Conjugate Gradient (CG) method in [Schubiger et al. 2020]. Specifically, the matrix-vector multiplication of each CG iteration is executed on the GPU, outperforming serial and parallel OSQP implementations on Central Processing Units (CPUs) for strongly convex-quadratic OCPs with approximately 10^5 decision variables or more. An adaptation of this approach to Field Programmable Gate Arrays (FPGAs) provides even further speedup [Wang et al. 2023]. GPU implementations of condensed interior point methods for linear-quadratic MPC and non-convex Alternating Current Optimal Power Flow (AC-OPF) are considered in [Cole et al. 2023] and [Pacaud et al. 2024; Shin et al. 2024], respectively. For AC-OPF problems with 200,000 decision variables, the latter demonstrates a speedup of approximately 10 over ipopt [Wächter and Biegler 2006] running on a CPU, showcasing the strong potential of tailored GPU implementations.

Where to: centralized or distributed MPC? In view of the above, it is not clear when DMPC indeed provides computational benefits over centralized MPC. A precursor to this question concerns the scalability of DMPC to large networks. Early simulation results show that, for the considered scenarios, the convergence speed of ADMM depends on the

- coupling strengh,
- number of connections between subsystems,
- and stability of the subsystem dynamics,

but not on the number of subsystems in the network [Conte et al. 2012]. A possible explanation for this is the *exponential decay of sensitvity* in graph-structured NLPs by which changes to a CPSoS have little effect on distant subsystems in the graph [Shin et al. 2022]. The same phenomenon is observed in simulations with energy networks [Behrunani et al. 2024]. There, the OCP is a mixed integer linear program and the addition of new subsystems to the network does not increase the number of required ADMM iterations. As a result, DMPC scales favorably compared to centralized MPC, even though centralized MPC is faster than DMPC for the considered use cases. A similar observation has been made for the nonlinear DMPC of coupled van der Pol oscillators [Huber et al. 2022], where the addition of new subsystems does not affect the per-subsystem ADMM execution time. The scalability of ADMM for linear-quadratic DMPC based on system level synthesis is investigated in [Alonso et al. 2023]. DMPC there shows better scalability than centralized MPC for a mesh of second-order systems with coupled swing dynamics, exhibiting a solve time which is proportional to the number of states in the centralized system. While the latter three studies show competitive scalability of DMPC compared to centralized MPC, they execute ADMM sequentially on a single machine and the reported time measurements only assume computations among subsystems could be parallelized. In contrast, a comparison between parallel implementations of ADMM, dual decomposition, and the interior point solver of CPLEX [IBM 2009] for QPs

is presented in [Kozma et al. 2015]. There, decentralized solvers were found *not* to compare well with CPLEX in terms of overall runtime.

Hence, existing results suggest DMPC has the potential to scale well, because the addition of more subsystems has little effect on the decentralized optimizer convergence. However, as there also has been tremendous algorithmic and computational progress for centralized algorithms, the potential for time savings of parallel DMPC over centralized implementations has not been fully explored yet. In particular, it is not clear at which scale these savings will be realized and whether employing a finite number of decentralized optimizer iterations will lead to a significant loss in control performance.

1.3 Thesis Outline and Contributions

Cooperative DMPC aims to reconcile the strong performance of centralized MPC with the scalability and resilience of distributed control. To this end, decentralized optimization algorithms are employed to solve a cooperative OCP online without requiring centralized computation. However, real-time requirements only allow for a finite number of optimizer iterations to be performed in each sampling interval, causing an inherent trade-off between cooperation and decentralization. The above summary on the state of the art of cooperative DMPC uncovered the following limitations of existing approaches:

1. Stability: Preceding stability guarantees for nonlinear cooperative DMPC either require feasible initializations—which are difficult to obtain in practice—or assume optimizer convergence to infer system stability.

2. Implementation: While prior hardware experiments showed promising results, the attainable sampling interval under distributed computation was limited to 100 ms. For ADMM, the computational complexity of solving Nonlinear Programs (NLPs) on each subsystem was found to be a bottleneck when running DMPC on embedded hardware.

3. Scalability: DMPC was found to scale well in numerical simulations, but a detailed comparison between decentralized methods and state-of-the-art centralized solvers appears to be missing. Thus, it is unclear when cooperative DMPC pays off from a computational point of view.

The remainder of this thesis addresses these limitations as follows.

Chapter 2—Optimization Algorithms for Cooperative Distributed Model Predictive Control

Chapter 2 summarizes the main classes of optimization algorithms available in the literature for solving cooperative OCPs. The chapter places particular emphasis on decentralized optimization

methods, i.e. algorithms which require no centralized coordinator. We begin with a discussion on decentralized convex optimization methods which has appeared in [Stomberg et al. 2022a] and we place particular emphasis on the Alternating Direction Method of Multipliers (ADMM), because of its widespread success in theory, simulation, and real-world implementations of cooperative DMPC. We then move on to non-convex NLPs where, compared to convex QPs, fewer decentralized methods with guarantees appear to be available. Notably, most algorithms with guaranteed convergence in the presence of non-convex constraints require subsystems to solve NLPs, which can complicate embedded control implementations.

Chapter 3—Decentralized Sequential Quadratic Programming

Chapter 3 addresses this gap and proposes a bi-level decentralized Sequential Quadratic Programming (dSQP) framework with convergence guarantees for non-convex constraints. Instead of solving NLPs, the subsystems only have to solve small-scale convex QPs, which—as we will see in Chapter 5—proves benefitial in embedded implementations. The method combines an inequality-constrained SQP scheme on the outer level with ADMM on the inner level for solving the arising QPs. Convergence to regular Karush-Kuhn-Tucker (KKT) points under early termination of ADMM within each SQP step is ensured through a tailored inexact Newton-type stopping criterion. This allows to view dSQP as an inexact Newton method, where ADMM decentralizes the computations among subsystems. The chapter concludes with a numerical example on Alternating Current-Optimal Power Flow (AC-OPF), demonstrating competitive performance of dSQP when compared to state-of-the-art algorithms. The core algorithmic idea, convergence analysis, and numerical results are based on the conference paper [Stomberg et al. 2022b]. Here, we additionally present a novel two-block dSQP variant with simplified convergence analysis under coupled inequality constraints.

Chapter 4—Decentralized Real-Time Iterations

Chapter 4 discusses the application of dSQP to nonlinear DMPC and presents the main theoretical contribution of the thesis. To address the real-time requirements in control and inspired by the success of Real-Time Iteration (RTI) schemes for centralized NMPC, we apply only one or few outer dSQP iterations per control step. Moreover, we remove the inexact Newton stopping criterion and instead fix the number of ADMM iterations on the inner level. This facilitates real-time execution and results in a decentralized RTI (dRTI) algorithm. We derive a novel bound on the inner ADMM iterations guaranteeing linear convergence of dSQP to regular KKT points. Then, we combine the linear dSQP convergence with stability guarantees for centralized RTIs from the literature and prove the local exponential stability of the combined system-optimizer dynamics in closed-loop. To the best of the author's knowledge, this is the first theoretical stability guarantee which studies both system *and* optimizer in cooperative nonlinear DMPC that does not require

centralized coordination or feasible optimizer initializations. In contrast, prior results from the literature assume the convergence of either system *or* optimizer to infer stability for the other. We conclude the chapter by testing dRTI in simulation for the swing up of a chain of coupled inverted pendulums. This chapter is based on the journal article [Stomberg et al. 2025a].

Chapter 5—Application to Multi-Robot Formation Control

Chapter 5 tests dRTI in hardware experiments with robot formations. The distributed software and computing environments implement neighbor-to-neighbor communication between machines, which allows to investigate the effect of communication delays on real-time execution. We first study the formation control of mobile robots and then continue with a team of hovercraft moving on an air hockey table. By testing a range of scenarios on two distinct hardware platforms, we showcase the strenghts, prospects, and current limitations of dRTI. We cover test cases with control sampling intervals between $\delta = 200\,\text{ms}$ and $\delta = 50\,\text{ms}$ and increase the difficulty from linear-quadratic DMPC via nonlinear DMPC with inter-robot collision avoidance to experiments in dynamic environments. The chapter presents the following two main contributions. First, our experiments with mobile robots appear to be the first hardware tests of a DMPC scheme which solves non-convex OCPs *and* enjoys nominal stability guarantees. Second, the experiments with hovercraft improve upon the state of the art by increasing the possible sampling frequency of embedded cooperative DMPC from 5 Hz to 20 Hz. The latter achievement is facilitated by the low onboard computation footprint of dSQP compared to nonlinear ADMM which requires to solve NLPs onboard, limiting the optimizer iterations and control sampling interval in prior embedded DMPC experiments [Van Parys and Pipeleers 2017]. The results demonstrate that dRTI can be real-time feasible in relevant control applications, even when subject to slow wireless communication. Analyses of the DMPC solve times reveal that both subsystem computations and neighbor-to-neighbor communication can be equally relevant, with communication taking the major share in experiments with wireless communication. The results of this chapter have been published in the journal article [Stomberg et al. 2023] and the conference paper [Stomberg et al. 2025c].

Chapter 6—Computational Performance for Large-Scale Optimal Control

Superior scalability compared to schemes with a central coordinator is often quoted as a main motivation for DMPC, alleviating the existence of a single computational bottleneck. However, comparisons between the numerical performance of centralized and distributed MPC appear to be scarce. Chapter 6 thus analyzes the scalability of dRTI by reference to numerical simulations for the frequency control of power grids. We compare the computational performance of dRTI to state-of-the-art interior point solvers for OCPs with up to $333,000$ decision variables. To this end, we test a multi-threaded dRTI implementation on a multi-core CPU cluster and, to provide a fair comparison, use multi-threaded linear algebra packages inside the centralized solvers. Our case

study investigates the effects of (i) the number of subsystems in the network, i.e. the total grid size; (ii) subsystem size, i.e. the number of grid buses per subsystem; (iii) subsystem difficulty, i.e. the number of loads per subsystem; and (iv) required coordination between subsystems, i.e. the distribution of loads and generators in the grid. The results show fast convergence of dRTI to suboptimal inputs, achieving good control performance in closed loop. Finally, the number of subsystems has little impact on the required dRTI iterations across a wide range of scenarios, indicating good scalability for large networks. This chapter is based on the conference paper [Stomberg et al. 2025b].

Chapter 7 summarizes the thesis and provides an outlook on future work.

2 Optimization Algorithms for Cooperative Distributed Model Predictive Control

This chapter surveys state-of-the-art methods for solving partially separable programs arising in cooperative DMPC. Section 2.2 discusses algorithms for solving partially separable convex QPs that appear in linear-quadratic DMPC and Section 2.3 then focuses on non-convex NLPs arising from nonlinear dynamics or otherwise. Section 2.4 summarizes key properties about the covered methods and presents a small-scale example to illustrate possible pitfalls arising from infeasible optimizer iterates.

The discussion on algorithms for linear-quadratic DMPC in Section 2.2, the summary of the essentially decentralized Interior Point (d-IP) method, and the numerical example have appeared in the survey [Stomberg et al. 2022a].[1] The essentially decentralized Active Set Method (d-ASM) was first presented in [Stomberg et al. 2021] and d-IP is from [Engelmann et al. 2021b].

2.1 Problem Statement

The cooperative OCP (1.3) to be solved in DMPC can be written as the partially separable NLP

$$\min_{z} \sum_{i \in \mathcal{S}} f_i(z_i) \tag{2.1a}$$

$$\text{subject to} \quad g_i(z_i) = 0 \mid \nu_i \quad \forall i \in \mathcal{S}, \tag{2.1b}$$

$$h_i(z_i) \leq 0 \mid \mu_i \quad \forall i \in \mathcal{S}, \tag{2.1c}$$

$$\sum_{i \in \mathcal{S}} E_i z_i = b \mid \lambda. \tag{2.1d}$$

Each subsystem $i \in \mathcal{S}$ is assigned a decision variable $z_i \in \mathbb{R}^{n_i}$, an objective $f_i : \mathbb{R}^{n_i} \to \mathbb{R}$, and constraints $g_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_{g_i}}$ and $h_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_{h_i}}$. We assume the functions f_i, g_i, and h_i are three times continuously differentiable for all $i \in \mathcal{S}$. The constraint (2.1d) with sparse matrices $E_i \in \mathbb{R}^{n_c} \times \mathbb{R}^{n_i}$ and $b \in \mathbb{R}^{n_c}$ couples neighboring subsystems. Notice that any coupled optimization problem may be formulated as (2.1) by introducing decision variables that are then coupled via (2.1d).

The notation in NLP (2.1) describes that $\nu_i \in \mathbb{R}^{n_{g_i}}$, $\mu_i \in \mathbb{R}^{n_{h_i}}$, and $\lambda \in \mathbb{R}^{n_c}$ are the Lagrange multipliers to the respective constraints. We denote the concatenation of vectors x and y into a column vector by (x, y), the vertical concatenation of matrices A and B by (A, B), and their horizontal concatenation by $[A, B]$. The centralized variables of NLP (2.1) are $z \doteq (z_1, \ldots, z_S) \in \mathbb{R}^n$, $\nu \doteq (\nu_1, \ldots, \nu_S) \in \mathbb{R}^{n_g}$, and $\mu \doteq (\mu_1, \ldots, \mu_S) \in \mathbb{R}^{n_h}$. The NLP can be written in centralized

[1] The open-access article [Stomberg et al. 2022a] has been published under the CC BY 4.0 license, cf. `https://creativecommons.org/licenses/by/4.0/`.

form (A.1) by setting

$$f(z) \doteq \sum_{i \in \mathcal{S}} f_i(z_i),\ g(z) \doteq (g_1(z_1), \dots, g_S(z_S)),\ h(z) \doteq (h_1(z_1), \dots, h_S(z_S)), \quad \text{and} \quad E \doteq [E_1 \dots E_S].$$

Assumption 2.1 (Linearly independent coupling constraints)**.** *The centralized coupling matrix E has full row rank.* □

Assumption 2.1 is a prerequisite for the Linear Indepence Constraint Qualification (LICQ) which we require in many of the following algorithms. Moreover, Assumption 2.1 ensures that constraint (2.1d) is feasible for all $b \in \mathbb{R}^{n_c}$. A crucial property of NLP (2.1) is the sparsity of the coupling matrices E_i which results from the underlying graph structure of the centralized OCP. All algorithms covered in this chapter exploit the sparsity of E to decentralize computations. As noted in [Engelmann et al. 2020], the coupling constraints (2.1d) generalize the widely adopted notion of *consensus problems* [Nedić and Liu 2018]. We detail the relation between consensus problems and the partially separable program (2.1) with the following two definitions. We denote the j-th row of a matrix A by $[A]_j$ and the component at position (i, j) by $[A]_{ij}$.

Definition 2.1 (Assigned coupling constraints and n-assignment [Engelmann et al. 2020, Definition 1])**.**
A subsystem $i \in \mathcal{S}$ is said to be assigned to coupling constraint $j \in \{1, \dots, n_c\}$, if $[E_i]_j \neq 0$. We collect all subsystems assigned to coupling constraint $j \in \{1, \dots, n_c\}$ in the set $\mathcal{S}(j) \doteq \{i \in \mathcal{S} \mid i \text{ is assigned to } j\}$. A coupling constraint $j \in \{1, \dots, n_c\}$ is said to be n-assigned, if $|\mathcal{S}(j)| = n$. Furthermore, the partially separable program (2.1) *is said to be n-assigned, if $|\mathcal{S}(j)| \leq n$ for all $j \in \{1, \dots, n_c\}$.*

Definition 2.2 (Consensus problem in partially separable form)**.** *The partially separable program* (2.1) *is said to be a consensus problem, if $b = 0$ and if, for all $j \in \{1, \dots, n_c\}$, there exists one element for which $[E]_{jo} = 1$, one element for which $[E]_{jc} = -1$, and if $[E]_{jl} = 0$ for all other elements $l \in \{1, \dots, n\}/\{o, c\}$.*

That is, we define consensus problems as 2-assigned partially separable programs where each row of E has two non-zero entries and where the right-hand side of (2.1d) is zero.

Rewriting an OCP as consensus problem in partially separable form The reader may wonder how we arrive at the partially separable NLP (2.1) coming from the cooperative OCP (1.3): A common procedure for reformulating the OCP is to replace, for all $i \in \mathcal{S}$, the in-neighbor state x_j by a copy w_{ji} for all $j \in \mathcal{N}_i^{\text{in}}$ when evaluating ℓ_i, f_i^δ, and the coupling constraint (1.3f) [El Fawal et al. 1998; Bestler and Graichen 2019; Stomberg et al. 2022a]. Define $w_i \doteq (w_{ji})_{j \in \mathcal{N}_i^{\text{in}}}$ and let $\boldsymbol{w}_i \doteq (w_i[0], \dots, w_i[N]) \in \mathbb{R}^{(N+1) \cdot n_i^{\text{in}}}$ denote the prediction by subsystem i for the state trajectories of in-neighbors. The decision variable of subsystem i then is $z_i \doteq (\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{w}_i)$ and the coupling constraints in (2.1d) match original and copied variables, i.e., they enforce the constraints

$$x_j[\tau] - w_{ji}[\tau] = 0 \quad \text{for all} \quad \tau \in \mathbb{I}_{[0,N]}, \quad j \in \mathcal{N}_i^{\text{in}}, \quad \text{and} \quad i \in \mathcal{S}.$$

Example 2.1 (Formulating a cooperative OCP as a partially separable NLP [Stomberg et al. 2022a])**.** *To illustrate the reformulation via state copies, consider three subsystems with coupled dynamics*

$$\begin{aligned} x_1[\tau+1] &= x_1[\tau] + u_1[\tau], \quad & x_1[0] &= x_1(t), \\ x_2[\tau+1] &= x_1[\tau] + x_2[\tau], \quad & x_2[0] &= x_2(t), \\ x_3[\tau+1] &= x_2[\tau] + x_3[\tau], \quad & x_3[0] &= x_3(t). \end{aligned}$$

By introducing the copies $w_2 \doteq x_1$ and $w_3 \doteq x_2$, we can rewrite the dynamics as

$$\begin{aligned} x_1[\tau+1] &= x_1[\tau] + u_1[\tau], \quad & x_1[0] &= x_1(t), \\ x_2[\tau+1] &= w_2[\tau] + x_2[\tau], \quad & x_2[0] &= x_2(t), \\ x_3[\tau+1] &= w_3[\tau] + x_3[\tau], \quad & x_3[0] &= x_3(t), \end{aligned}$$

For $N = 1$, the decision variables are $z_1 = (x_1[0], x_1[1], u_1[0])$, $z_2 = (x_2[0], x_2[1], w_2[0])$, and $z_3 = (x_3[0], x_3[1], w_3[0])$. The coupling between original and copied states then reads

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{E_1} z_1 + \underbrace{\begin{bmatrix} 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix}}_{E_2} z_2 + \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}}_{E_3} z_3 = 0.$$

□

Distributed, decentralized, and essentially decentralized algorithms

We categorize optimization algorithms based on their communication architecture. *Distributed* optimization algorithms decompose expensive computations among the subsystems, but still perform some computations centrally [Bertsekas and Tsitsiklis 1989]. *Decentralized* optimization, in contrast, does not require a central coordinator and instead exclusively relies on neighbor-to-neighbor communication [Nedić et al. 2018]. As a middle ground, *essentially decentralized* algorithms perform computationally simple steps with network-wide communication but without centralized computation [Engelmann et al. 2021b]. Figure 2.1 links the MPC architectures proposed in [Scattolini 2009] and discussed in Chapter 2 to the corresponding types of optimization algorithms. Centralized MPC solves the OCP with centralized optimization methods, hierarchical MPC employs distributed optimization, and cooperative DMPC relies on decentralized optimization algorithms.

2.2 Algorithms for Linear-Quadratic DMPC

An important class of optimization problems for DMPC are partially separable QPs which arise if the system dynamics are linear, the constraints are polyhedral, and the cost functions are quadratic,

$$\min_z \sum_{i \in \mathcal{S}} \frac{1}{2} z_i^\top H_i z_i + a_i^\top z_i \tag{2.2a}$$

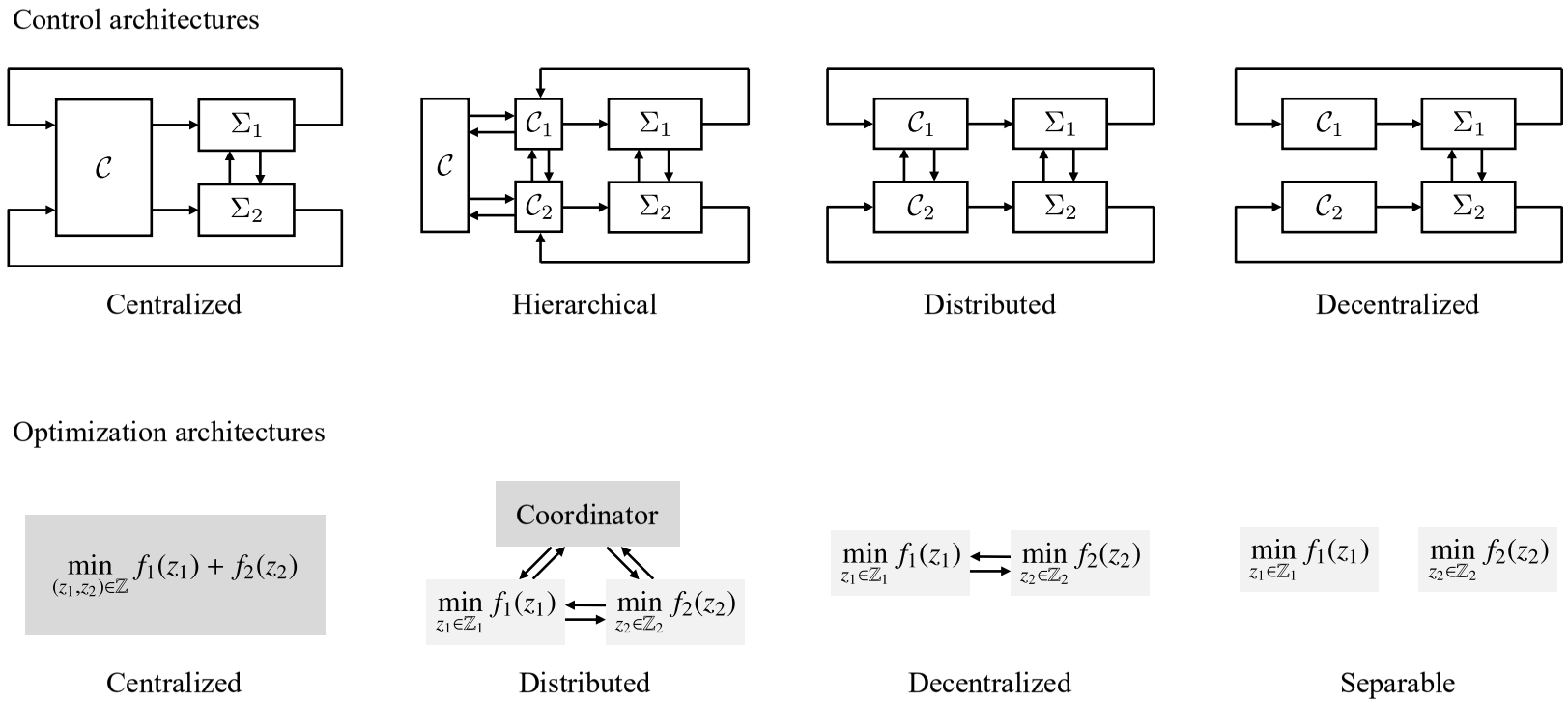


Figure 2.1: Optimization and control architectures for CPSoS. This thesis discusses distributed MPC based on decentralized optimization. Control architecture diagrams inspired by [Scattolini 2009] and figure adapted from [Stomberg et al. 2023].

$$\text{subject to} \quad A_i^{\mathrm{E}} z_i = b_i^{\mathrm{E}} \mid \nu_i \quad \forall i \in \mathcal{S}, \tag{2.2b}$$

$$A_i^{\mathrm{I}} z_i \leq b_i^{\mathrm{I}} \mid \mu_i \quad \forall i \in \mathcal{S}, \tag{2.2c}$$

$$\sum_{i \in \mathcal{S}} E_i z_i = b \mid \lambda. \tag{2.2d}$$

Here, $a_i \in \mathbb{R}^{n_i}$, $A_i^{\mathrm{E}} \in \mathbb{R}^{n_{g_i} \times n_i}$, and $A_i^{\mathrm{I}} \in \mathbb{R}^{n_{h_i} \times n_i}$ for all $i \in \mathcal{S}$. The Hessians $H_i \in \mathbb{R}^{n_i \times n_i}$ are typically positive semidefinite or positive definite for all $i \in \mathcal{S}$ and the convexity requirements differ among algorithms, with positive definite Hessians usually allowing for faster convergence guarantees. Observe that copied state variables also have to be assigned weights in (2.2a) if H_i is required to be positive definite. For some algorithms, it is convenient to rewrite QP (2.2) as

$$\min_{z_1 \in \mathbb{Z}_1^{\mathrm{QP}}, \ldots, z_S \in \mathbb{Z}_S^{\mathrm{QP}}} \sum_{i \in \mathcal{S}} f_i^{\mathrm{QP}}(z_i) \quad \text{subject to} \quad \sum_{i \in \mathcal{S}} E_i z_i = b \mid \lambda,$$

where

$$f_i^{\mathrm{QP}}(z_i) \doteq \frac{1}{2} z_i^\top H_i z_i + a_i^\top z_i \quad \text{and} \quad \mathbb{Z}_i^{\mathrm{QP}} \doteq \left\{ z_i \in \mathbb{R}^{n_i} \;\middle|\; \begin{array}{l} A_i^{\mathrm{E}} z_i = b_i^{\mathrm{E}} \\ A_i^{\mathrm{I}} z_i \leq b_i^{\mathrm{I}} \end{array} \right\}.$$

Likewise, we write the subsystem constraints in centralized form as $\mathbb{Z}^{\mathrm{QP}} \doteq \mathbb{Z}_1^{\mathrm{QP}} \times \cdots \times \mathbb{Z}_S^{\mathrm{QP}}$.

2.2.1 Dual Decomposition

We define the Lagrangian $L : \mathbb{R}^n \times \mathbb{R}^{n_c} \to \mathbb{R}$ and dual function $q : \mathbb{R}^{n_c} \to \mathbb{R}$ to QP (2.2) as

$$L(z, \lambda) \doteq \sum_{i \in \mathcal{S}} \left(f_i^{\mathrm{QP}}(z_i) + \lambda^\top E_i z_i \right) - \lambda^\top b \quad \text{and} \quad q(\lambda) \doteq \min_{z \in \mathbb{Z}^{\mathrm{QP}}} L(z, \lambda).$$

In its simplest form, Dual Decomposition (DD) solves QP (2.2) by applying gradient ascent to the dual problem

$$\max_{\lambda} q(\lambda). \tag{2.3}$$

For all $i \in \mathcal{S}$, let f_i^{QP} be strongly convex with convexity parameter μ_{c} as defined in Appendix A.3 and let the set $\mathbb{Z}_i^{\mathrm{QP}}$ be convex and compact. Then, QP (2.2) satisfies strong duality [Bertsekas 2016, Proposition 6.2.2]. Moreover, $q(\lambda)$ is continuously differentiable and the dual gradient reads [Bertsekas 2016, Proposition 7.1.1]

$$\nabla q(\lambda) = \sum_{i \in \mathcal{S}} E_i z_i^{\lambda, \star} - b \quad \text{where} \quad z_i^{\lambda, \star} \doteq \operatorname*{argmin}_{z_i \in \mathbb{Z}_i^{\mathrm{QP}}} f_i^{\mathrm{QP}}(z_i) + \lambda^\top E_i z_i \quad \text{for all } i \in \mathcal{S}.$$

The gradient ascent update with step size $c > 0$ reads

$$\lambda^{k+1} = \lambda^k + c \nabla q(\lambda^k)$$

and convergence to the unique optimum is guaranteed if $c \in (0, 2/L_q)$, where $L_q > 0$ is a Lipschitz constant of $\nabla q(\lambda)$ [Bertsekas 2016, Proposition 1.2.2]. Here, the smallest Lipschitz constant of ∇q is [Richter et al. 2011, Theorem 4]

$$L_q^\star = \left\| E H^{-\frac{1}{2}} \right\|_2^2, \tag{2.4}$$

where $H \doteq \operatorname{diag}(H_1, \ldots, H_S)$ is a block diagonal matrix with block H_i at block position (i, i). Reformulating (2.4) yields

$$L_q^\star = \left\| E H^{-\frac{1}{2}} \right\|_2^2 \leq \|E\|_2^2 \cdot \left\| H^{-\frac{1}{2}} \right\|_2^2 = \|E\|_2^2 \cdot \left(\sigma_{\max}(H^{-\frac{1}{2}}) \right)^2 = \|E\|_2^2 \cdot \lambda_{\max}\left(H^{-1}\right) = \frac{\|E\|_2^2}{\mu_c}$$

and thus DD converges for any $c \in (0, 2\mu_c/\|E\|_2^2)$ [Braun and Grüne 2018]. The DD iterations read

$$z_i^k = \operatorname*{argmin}_{z_i \in \mathbb{Z}_i^{\mathrm{QP}}} f_i^{\mathrm{QP}}(z_i) + \lambda^{k\top} E_i z_i \quad \forall i \in \mathcal{S} \tag{2.5a}$$

$$\lambda^{k+1} = \lambda^k + c\left(\sum_{i \in \mathcal{S}} E_i z_i^k - b \right). \tag{2.5b}$$

The step (2.5a) requires each subsystem to solve a small-scale strongly convex QP and the update (2.5b) can be decentralized with the following three steps: First, note that (2.5) yields

$$z_i^{k+1} = \operatorname*{argmin}_{z_i \in \mathbb{Z}_i^{\mathrm{QP}}} f_i^{\mathrm{QP}}(z_i) + \left(\lambda^{k\top} + c\left(\sum_{j \in \mathcal{S}} z_j^{k\top} E_j^\top - b^\top \right) \right) E_i z_i \quad \forall i \in \mathcal{S}$$

Crucially, if $j \neq \mathcal{N}_i \cup \{i\}$, then $E_j^\top E_i = 0$ and hence

$$z_i^{k+1} = \operatorname*{argmin}_{z_i \in \mathbb{Z}_i^{\mathrm{QP}}} f_i^{\mathrm{QP}}(z_i) + \left(\lambda^{k\top} + c\left(\sum_{j \in \mathcal{N}_i \cup \{i\}} z_j^{k\top} E_j^\top - b^\top \right) \right) E_i z_i \quad \forall i \in \mathcal{S}.$$

That is, each subsystem only needs to receive the iterates z_j^k from its neighbors, but not from subsystems outside its neighborhood. Second, let each subsystem i store a local version $\lambda_i \in \mathbb{R}^{n_c}$ of λ satisfying

$$\lambda^\top E_i = \lambda_i^\top E_i \quad \forall i \in \mathcal{S}. \tag{2.6}$$

That is, λ_i includes those multipliers which correspond to coupling constraints that affect subsystem i. Third, the product $E_j^\top E_i$ is sparse for all $j \in \mathcal{N}_i \cup \{i\}$ and subsystem i only requires coupled components $[z_j^k]_p$ for all

$$p \in \left\{ \mathbb{I}_{[0,n_j]} \;\middle|\; \left[E_j^\top E_i \right]_p \neq 0 \right\} \tag{2.7}$$

of subsystem j to compute λ_i^{k+1}. The decentralized implementation of DD is given in Algorithm 2.1 which maintains (2.6) in Step 6.

Dual Decomposition With Fast Gradients

The classical DD Algorithm 2.1 applies gradient ascent to the dual problem (2.3) and exploits the sparsity of the coupling matrices E_i to decentralize the gradient calculation. This decomposition for ∇q can applied to a wide range of first-order methods, including *Fast Gradient Methods* (FGMs) [Necoara and Suykens 2008; Nesterov 2018]. For instance, a Nesterov-type acceleration is used for DD in [Conte et al. 2012],

$$\lambda^{k+1} = \bar{\lambda}^k + \frac{1}{L_q} \nabla q(\bar{\lambda}^k) \tag{2.8a}$$

Algorithm 2.1 Decentralized DD for solving (2.2) [Braun and Grüne 2018].

1: Initialization: $k = 0, \lambda^0, \{\lambda_i^0\}_{i\in\mathcal{S}}$ satisfying (2.6), $k_{\max}$, c
2: **while** $k < k_{\max}$ **do**
3: $\quad z_i^k = \underset{z_i\in\mathbb{Z}_i}{\operatorname{argmin}} f_i^{\mathrm{QP}}(z_i) + \lambda_i^{k\top} E_i z_i$ for all $i \in \mathcal{S}$.
4: $\quad$ Send $[z_i^k]_p$ for all $p \in \left\{\mathbb{I}_{[0,n_i]} \;\middle|\; \left[E_i^\top E_j\right]_p \neq 0\right\}$ to all $j \in \mathcal{N}_i$ and for all $i \in \mathcal{S}$.
5: $\quad$ Receive $[z_j^k]_p$ for all p satisfying (2.7) from all $j \in \mathcal{N}_i$ and for all $i \in \mathcal{S}$.
6: $\quad \lambda_i^{k+1} = \lambda_i^k + c\left(\sum_{j\in\mathcal{N}_i\cup\{i\}} E_i z_j^k - b\right)$ for all $i \in \mathcal{S}$.
7: $\quad k \leftarrow k + 1$
8: **end while**

$$\alpha^{k+1} = \frac{\alpha^k}{2}\left(\sqrt{(\alpha^k)^2 + 4} - \alpha^k\right) \tag{2.8b}$$

$$\beta^k = \frac{\alpha^k(1-\alpha^k)}{(\alpha^k)^2 + \alpha^{k+1}} \tag{2.8c}$$

$$\bar{\lambda}^{k+1} = \lambda^{k+1} + \beta^k\left(\lambda^{k+1} - \lambda^k\right). \tag{2.8d}$$

Here, $\bar{\lambda} \in \mathbb{R}^{n_c}$ is an intermediate iterate and $\beta > 0$ is the step size. The fastest convergence can be obtained by inserting the smallest Lipschitz constant $L_q^\star$ from (2.4) into the update (2.8a) [Richter et al. 2011]. Algorithm 2.2 summarizes a decentralized version using this fast gradient update.

Assumptions, convergence, feasibility, and termination Strong convexity of the objective and compactness of the constraint set together guarantee differentiability of the dual function and thus well-definedness of DD and DD-FGM [Bertsekas 2016, Proposition 7.1.1]. The convergence in $q(\lambda^\star) - q(\lambda^k)$ is guaranteed to be at least sublinear with rates $O(1/k)$ for DD [Nesterov 2018, Theorem 2.1.14] and $O(1/k^2)$ for DD-FGM [Richter 2012, Equation 9.10]. The iterates z_i^k are feasible with respect to $\mathbb{Z}_i^{\mathrm{QP}}$, but the coupling constraints (2.2d) are only satisfied asymptotically. Stopping criteria usually specify tolerances for $\nabla q(\lambda)$, i.e., the coupling constraint residual.

Closed-loop stability, warm-starting, and communication If the subsystems are only coupled via the cost functions, then feasibility of z_i^k with respect to the subsystem constraints guarantees closed-loop stability for appropriately designed terminal ingredients. However, if coupling is also present in the dynamics or system constraints, then infeasibility with respect to the coupling constraints (2.2d) must be considered. Constraint tightening can guarantee closed-loop stability through tailored stopping criteria or for sufficiently many optimizer iterations per control step [Giselsson and Rantzer 2014; Köhler et al. 2019]. DD is warm-started with a guess $\lambda^0 \in \mathbb{R}^{n_c}$ and thus does not require a feasible initialization. A benefit over an initialization $z^0 \in \mathbb{R}^n$ is that the coupling dimension n_c is typically much smaller than n. This can be exploited to obtain warm-starts through supervised learning [Chanfreut et al. 2021a]. The sparsity of the coupling matrices

Algorithm 2.2 Decentralized DD with fast gradients [Conte et al. 2012].

Initialization: $k = 0, \lambda^0, \{\lambda_i^0\}_{i\in\mathcal{S}}$ satisfying (2.6), $\bar{\lambda}_i^0 = \lambda_i^0$ for all $i \in \mathcal{S}$, $\alpha^0 = (\sqrt{5}-1)/2$, $k_{\max}$, L_q
while $k < k_{\max}$ **do**
  $z_i^k = \underset{z_i \in \mathbb{Z}_i^{\mathrm{QP}}}{\operatorname{argmin}} f_i^{\mathrm{QP}}(z_i) + \bar{\lambda}_i^{k\top} E_i z_i$ for all $i \in \mathcal{S}$.
  Send $[z_i^k]_p$ for all $p \in \left\{ \mathbb{I}_{[0,n_i]} \;\middle|\; \left[E_i^\top E_j\right]_p \neq 0 \right\}$ to all $j \in \mathcal{N}_i$ and for all $i \in \mathcal{S}$.
  Receive $[z_j^k]_p$ for all p satisfying (2.7) from all $j \in \mathcal{N}_i$ and for all $i \in \mathcal{S}$.
  $\lambda_i^{k+1} = \bar{\lambda}_i^k + \frac{1}{L_q}\left(\sum_{j\in\mathcal{N}_i\cup\{i\}} E_i z_j^k - b \right)$ for all $i \in \mathcal{S}$.
  $\alpha^{k+1} = \frac{\alpha^k}{2}\left(\sqrt{(\alpha^k)^2 + 4} - \alpha^k \right)$.
  $\beta^k = \frac{\alpha^k\left(1-\alpha^k\right)}{(\alpha^k)^2 + \alpha^{k+1}}$.
  $\bar{\lambda}_i^{k+1} = \lambda_i^{k+1} + \beta^k\left(\lambda_i^{k+1} - \lambda_i^k\right)$ for all $i \in \mathcal{S}$.
  $k \leftarrow k + 1$
end while

E_i allows to decentralize DD and Algorithms 2.1 and 2.2 exchange $2n_c$ floats per iteration between neighbors.

Extensions and applications Similar to most other methods covered in this chapter, DD is a family of algorithms instead of a specific method and comes with different flavors in terms of problem formulation and employed first order updates. For instance, it is possible to not only dualize the coupled equality constraints (2.2d) of the partially separable QP, but to dualize coupled inequality constraints directly [Giselsson et al. 2013]. In this case, a projected gradient method is applied to the dual problem to ensure non-negativity of the Lagrange multipliers to the coupled inequality constraints. Exemplary applications of DMPC based on DD include building automation, microgrid control, and water distribution networks [Trnka et al. 2011; Velasquez et al. 2019; Razzanelli et al. 2020; Braun et al. 2020; Lefebure et al. 2022].

2.2.2 Alternating Direction Method of Multipliers

ADMM is an augmented Lagrangian-based decomposition framework which is applicable to a wide range of optimization problems, including non-convex NLPs. We here present a *two-block ADMM* variant whose application to DMPC dates back at least to [Summers and Lygeros 2012; Conte et al. 2012].

In general, ADMM has received widespread attention for both linear-quadratic DMPC *and* nonlinear DMPC, see e.g. [Summers and Lygeros 2012; Conte et al. 2012; Rostami et al. 2017; Bestler and Graichen 2019; Tang and Daoutidis 2019; Burk et al. 2021b; Alonso et al. 2023]. However, to

the best of our knowledge, convergence guarantees for the decentralized variant of ADMM which we discuss here have only been established if the subsystem constraint sets $\mathbb{Z}_i$ are convex. Thus, we dicuss ADMM in this section on linear-quadratic DMPC. Nonetheless, we present the ADMM iterations for NLPs like (2.1)—and not only for QPs—to reflect nonlinear DMPC schemes based on ADMM as well. We comment on the difficulties for non-convex constraint sets at the end of this section.

To apply ADMM, we rewrite NLP (2.1) as the *two-block NLP*

$$\min_{y,z} \sum_{i\in\mathcal{S}} f_i(y_i) \tag{2.9a}$$

$$\text{subject to} \quad y_i \in \mathbb{Z}_i \qquad \forall i \in \mathcal{S}, \tag{2.9b}$$

$$z \in \mathbb{E}, \tag{2.9c}$$

$$y_i - z_i = 0 \mid \gamma_i \quad \forall i \in \mathcal{S}, \tag{2.9d}$$

where $y \doteq (y_1, \dots, y_S) \in \mathbb{R}^n$ and $\gamma \doteq (\gamma_1, \dots, \gamma_S) \in \mathbb{R}^n$. The subsystem constraint sets $\mathbb{Z}_i$ and the coupling constraint set $\mathbb{E}$ are defined as

$$\mathbb{Z}_i \doteq \left\{ y_i \in \mathbb{R}^{n_i} \;\middle|\; \begin{matrix} g_i(y_i) = 0 \\ h_i(y_i) \leq 0 \end{matrix} \right\} \text{ and } \mathbb{E} \doteq \left\{ z \in \mathbb{R}^n \;\middle|\; \sum_{i\in\mathcal{S}} E_i z_i = b \right\}.$$

NLP (2.9) owes its name to the two blocks of decision variables, y and z. Both blocks are coupled through the *consensus constraint* (2.9d). We define the augmented Lagrangian to NLP (2.9) as

$$L_\rho(y, z, \gamma) \doteq \sum_{i\in\mathcal{S}} L_{\rho,i}(y_i, z_i, \gamma_i) \doteq \sum_{i\in\mathcal{S}} \left(f_i(y_i) + \gamma_i^\top (y_i - z_i) + \frac{\rho}{2} \|y_i - z_i\|_2^2 \right).$$

The ADMM iterations for solving NLP (2.9) are given by

$$y_i^{k+1} = \operatorname*{argmin}_{y_i \in \mathbb{Z}_i} L_{\rho,i}(y_i, z_i^k, \gamma_i^k) \qquad \forall i \in \mathcal{S} \tag{2.10a}$$

$$z^{k+1} = \operatorname*{argmin}_{z\in\mathbb{E}} L_\rho(y^{k+1}, z, \gamma^k) \tag{2.10b}$$

$$\gamma_i^{k+1} = \gamma_i^k + \rho(y_i^{k+1} - z_i^{k+1}) \qquad \forall i \in \mathcal{S}. \tag{2.10c}$$

The updates (2.10a) for the first block of decision variables y and (2.10c) for the Lagrange multipliers γ trivially decompose between the subsystems because of the partially separable structure of NLP (2.9). We next discuss how the z-update (2.10b) can be implemented as a decentralized averaging step for consensus problems [Boyd et al. 2011, Chapter 7]. To this end, we first derive some technical results on the ADMM averaging step (2.10b).

Lemma 2.1 (Algebraic expression of the ADMM averaging step)**.** *Consider the two-block NLP* (2.9)*, suppose Assumption 2.1 holds, and define the averaging matrix*

$$M_{\text{avg}} \doteq I - E^\top (EE^\top)^{-1} E.$$

Then, the following holds for the iterates $\left\{\left(y^k, z^k, \gamma^k\right)\right\}$ *produced by ADMM:*

Algorithm 2.3 ADMM for NLP (2.1).

1: Initialization: $k = 0, \lambda^0, \gamma_i^0 = E_i^\top \lambda^0, z_i^0$ for all $i \in \mathcal{S}$, $k_{\max}, \rho$
2: **while** $k < k_{\max}$ **do**
3: $\quad y_i^{k+1} = \underset{y_i \in \mathbb{Z}_i}{\operatorname{argmin}} L_{\rho,i}(y_i, z_i^k, \gamma_i^k)$ for all $i \in \mathcal{S}$.
4: $\quad z^{k+1} = \underset{z \in \mathbb{B}}{\operatorname{argmin}} L_\rho(y^{k+1}, z, \gamma^k)$.
5: $\quad \gamma_i^{k+1} = \gamma_i^k + \rho(y_i^{k+1} - z_i^{k+1})$ for all $i \in \mathcal{S}$.
6: $\quad k \leftarrow k + 1$
7: **end while**

i) If $M_{\text{avg}}\gamma^k = 0$ and if $b = 0$, then $z^{k+1} = M_{\text{avg}} y^{k+1}$.

ii) If $M_{\text{avg}}\gamma^k = 0$ and if $b = 0$, then $M_{\text{avg}}\gamma^{k+1} = 0$.

iii) If $\gamma^0 \doteq E^\top \lambda^0$ for any $\lambda^0 \in \mathbb{R}^{n_c}$, then $M_{\text{avg}}\gamma^0 = 0$.

iv) The Lagrange multipliers satisfy $\gamma^{k+1} = E^\top \lambda^{k+1}$ for all $k \geq 0$.

Proof. i) The z-update (2.10b) requires to solve a strongly convex equality-constrained QP and thus can be solved in closed form by solving the KKT system

$$\begin{bmatrix} \rho I & E^\top \\ E & 0 \end{bmatrix} \begin{bmatrix} z^{k+1} \\ \lambda^{k+1} \end{bmatrix} = \begin{bmatrix} \rho y^{k+1} + \gamma^k \\ 0 \end{bmatrix}. \tag{2.11}$$

This yields

$$z^{k+1} = \underbrace{\left(I - E^\top (EE^\top)^{-1} E\right)}_{M_{\text{avg}}} \left(y^{k+1} + \frac{\gamma^k}{\rho}\right) = M_{\text{avg}} y^{k+1} + \underbrace{M_{\text{avg}} \frac{\gamma^k}{\rho}}_{0},$$

where $EE^\top$ is invertible because E has full row rank.

ii) We begin by proving

$$\begin{aligned} M_{\text{avg}}^2 &= (I - E^\top (EE^\top)^{-1} E)(I - E^\top (EE^\top)^{-1} E) \\ &= I - 2E^\top (EE^\top)^{-1} E + E^\top \underbrace{(EE^\top)^{-1} EE^\top}_{I} (EE^\top)^{-1} E \\ &= I - E^\top (EE^\top)^{-1} E = M_{\text{avg}}. \end{aligned}$$

The γ-update (2.10c) yields

$$\begin{aligned} M_{\text{avg}}\gamma^{k+1} &= M_{\text{avg}}\gamma^k + \rho M_{\text{avg}}(y^{k+1} - z^{k+1}) \\ &= \rho M_{\text{avg}}(y^{k+1} - M_{\text{avg}} y^{k+1}) \\ &= \rho(M_{\text{avg}} y^{k+1} - M_{\text{avg}}^2 y^{k+1}) \\ &= \rho(M_{\text{avg}} y^{k+1} - M_{\text{avg}} y^{k+1}) = 0. \end{aligned}$$

iii) Let $\gamma^0 \doteq E^\top \lambda^0$. Then

$$M_{\text{avg}}\gamma^0 = M_{\text{avg}}E^\top\lambda^0 = (I - E^\top(EE^\top)^{-1}E)E^\top\lambda^0 = E^\top\lambda^0 - E^\top \underbrace{(EE^\top)^{-1}EE^\top}_{I}\lambda^0 = 0.$$

iv) The γ-update (2.10c) and the top block rows in the KKT conditions (2.11) yield

$$\begin{aligned}\gamma^{k+1} &= \gamma^k + \rho(y^{k+1} - z^{k+1})\\ &= \gamma^k + \rho y^{k+1} - \rho z^{k+1}\\ &= \gamma^k + \rho y^{k+1} - (\rho y^{k+1} + \gamma^k - E^\top\lambda^{k+1})\\ &= E^\top\lambda^{k+1}.\end{aligned} \tag{2.12}$$

■

If NLP (2.1) is a consensus problem, then the update $z^{k+1} = M_{\text{avg}}y^{k+1}$ can be efficiently evaluated as an averaging step with decentralizd communication. We next adapt the discussion on the averaging step from [Boyd et al. 2011, Chapter 7] to the two-block NLP (2.9).

Remark 2.1 (Averaging step in ADMM for consensus problems)**.** *By Definition 2.2, each coupling constraint $j \in \{1,\dots,n_c\}$ in a consensus problem enforces the equivalence between two components of the centralized decision vector z. These two components can be regarded as two instances of a global variable $x_g \in \mathbb{R}$. The consensus constraint j is met if the two instances hold the same numerical value for x_g.*

We denote the index set of the global variables in a consensus problem by $\mathbb{G} \doteq \{1,\dots,n-n_c\}$. For all $g \in \mathbb{G}$ and for all $i \in \mathcal{S}$, we collect the indices of the components of z_i that are instances of x_g in the set $\mathcal{I}_{g_i} \doteq \left\{p \in \{1,\dots,n_i\} \;\middle|\; [z_i]_p = x_g\right\}$. For all indices $g \in \mathbb{G}$, we collect the subsystems that hold an instance of x_g in the set $\mathcal{S}_g \doteq \left\{i \in \mathcal{S} \;\middle|\; \mathcal{I}_{g_i} \neq \emptyset\right\}$. The z-update (2.10b) *then reads*

$$\min_{z,\{x_g\},g\in\mathbb{G}} \sum_{g\in\mathbb{G}}\sum_{i\in\mathcal{S}_g}\sum_{p\in\mathcal{I}_{g_i}} \left(\frac{\rho}{2}[z_i]_p^2 - \rho[y_i^{k+1}]_p[z_i]_p - [\gamma_i^k]_p[z_i]_p\right) \tag{2.13a}$$

$$\text{subject to} \quad x_g = [z_i]_p \quad \forall p \in \mathcal{I}_{g_i}, \quad \forall i \in \mathcal{S}_g, \quad \forall g \in \mathbb{G}. \tag{2.13b}$$

By resolving the consensus constraint (2.13b)*, this is equivalent to the unconstrained QP*

$$\min_{\{x_g\},g\in\mathbb{G}} \sum_{g\in\mathbb{G}}\sum_{i\in\mathcal{S}_g}\sum_{j\in\mathcal{I}_{g_i}} \left(\frac{\rho}{2}x_g^2 - \rho[y_i^{k+1}]_j x_g - [\gamma_i^k]_j x_g\right).$$

Solving the conditions of optimality for x_g yields

$$x_g^{k+1} = \frac{1}{N_g}\sum_{i\in\mathcal{S}_g}\sum_{p\in\mathcal{I}_{g_i}} \left([y_i^{k+1}]_p + [\gamma_i^k]_p/\rho\right) \quad \forall g \in \mathbb{G}, \tag{2.14}$$

where $N_g \doteq \sum_{i\in\mathcal{S}_g}|\mathcal{I}_{g_i}|$ is the total number of instances of x_g. Hence, if $\sum_{i\in\mathcal{S}_g}\sum_{p\in\mathcal{I}_{g_i}}[\gamma_i^k]_p = 0$ for all $g \in \mathbb{G}$, then the update is given by

$$x_g^{k+1} = \frac{1}{N_g}\sum_{i\in\mathcal{S}_g}\sum_{p\in\mathcal{I}_{g_i}} [y_i^{k+1}]_p \quad \forall g \in \mathbb{G}. \tag{2.15}$$

That is, the update x_g^{k+1} is the average value of all instances stored in the vector y^{k+1} that correspond to the scalar x_g. The averaging step $z^{k+1} = M_{\text{avg}} y^{k+1}$ can hence be implemented in decentralized fashion by completing the following three steps for all $g \in \mathbb{G}$:

1. *For all $i \in \mathcal{S}_g$ and for all $p \in \mathcal{I}_{g_i}$, communicate $[y_i^{k+1}]_p$ to a subsystem $s \in \mathcal{S}$.*
2. *Let subsystem s evaluate the average* (2.15) *to compute x_g^{k+1}.*
3. *Communicate x_g^{k+1} to all $i \in \mathcal{S}_g$ and set $[z_i^{k+1}]_p = x_g^{k+1}$ for all $p \in \mathcal{I}_{g_i}$ and for all $i \in \mathcal{S}_g$.* □

Remark 2.2 (Decentralized implementation of ADMM for DMPC)**.** *Consider a reformulation of the cooperative OCP* (1.3) *as a partially separable NLP* (2.1) *through the introduction of copied state variables as illustrated in Example 2.1. The global variables then are the predicted centralized state and input trajectories $\boldsymbol{x}$ and $\boldsymbol{u}$. For all $i \in \mathcal{S}$, we have*

$$\mathcal{S}_{[x_i[\tau]]_p} = \mathcal{N}_i^{\text{out}} \cup \{i\} \quad \text{for all} \quad p \in \{1, \dots, n_{x_i}\} \quad \text{and for all} \quad \tau \in \mathbb{I}_{[0,N]}.$$

That is, only subsystem i and its out-neighbors hold instances of the predicted state trajectory $\boldsymbol{x}_i$. Specifically, subsystem i optimizes over the original trajectory $\boldsymbol{x}_i$ and each out-neighbor $j \in \mathcal{N}_i^{\text{out}}$ optimizes over the copy trajectory $\boldsymbol{w}_{ij}$, i.e., each subsystem in the set $\mathcal{N}_i^{\text{out}} \cup \{i\}$ holds one instance of each global decision variable in $\boldsymbol{x}_i$. In contrast, predicted input trajectories are not shared in this formulation such that only subsystem i optimizes over $\boldsymbol{u}_i$.

For all $i \in \mathcal{S}$, we partition the two ADMM decision variable blocks as $y_i \doteq (\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{w}_i)$ and $z_i \doteq (\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{u}}_i, \bar{\boldsymbol{w}}_i)$ such that $\bar{\boldsymbol{x}}_i = \boldsymbol{x}_i$, $\bar{\boldsymbol{u}}_i = \boldsymbol{u}_i$, and $\bar{\boldsymbol{w}}_i = \boldsymbol{w}_i$ upon convergence. If $M_{\text{avg}}\gamma^k = 0$, then the z-update can be implemented with neighbor-to-neighbor communication:

1. *For all $i \in \mathcal{S}$, send $\boldsymbol{w}_{ji}^{k+1}$ to all $j \in \mathcal{N}_i^{\text{in}}$ and receive $\boldsymbol{w}_{ij}^{k+1}$ from all $j \in \mathcal{N}_i^{\text{out}}$.*
2. *Let each subsystem $i \in \mathcal{S}$ compute the average $\bar{\boldsymbol{x}}_i^{k+1} = \dfrac{1}{|\mathcal{N}_i^{\text{out}}| + 1}\left(\boldsymbol{x}_i^{k+1} + \displaystyle\sum_{j \in \mathcal{N}_i^{\text{out}}} \boldsymbol{w}_{ij}^{k+1}\right)$.*
3. *For all $i \in \mathcal{S}$, send $\bar{\boldsymbol{x}}_i^{k+1}$ to all $j \in \mathcal{N}_i^{\text{out}}$ and receive $\bar{\boldsymbol{x}}_j^{k+1}$ from all $j \in \mathcal{N}_i^{\text{in}}$.*
4. *For all $i \in \mathcal{S}$, set $\bar{\boldsymbol{u}}_i^{k+1} = \boldsymbol{u}_i^{k+1}$ and $\bar{\boldsymbol{w}}_{ji}^{k+1} = \bar{\boldsymbol{x}}_j^{k+1}$ for all $j \in \mathcal{N}_i^{\text{in}}$.*
5. *For all $i \in \mathcal{S}$, stack $\bar{\boldsymbol{w}}_i^{k+1} = (\bar{\boldsymbol{w}}_{ji}^{k+1})_{j \in \mathcal{N}_i^{\text{in}}}$ and form $z_i^{k+1} = (\bar{\boldsymbol{x}}_i^{k+1}, \bar{\boldsymbol{u}}_i^{k+1}, \bar{\boldsymbol{w}}_i^{k+1})$.* □

Assumptions, convergence, feasibility, and termination The consensus dual variable γ is guaranteed to converge to a solution $\gamma^\star$ and consensus is achieved in the limit, if the objectives f_i are convex, closed, and proper[2], if the constraints $\mathbb{Z}_i$ are convex, and if a solution exists for NLP (2.1) [Boyd et al. 2011, Chapter 3], cf. [Bertsekas 2016, Proposition 6.1.6]. For more specific

[2] See Appendix A.3 for a definition of convex, closed, and proper functions.

convex problems such as QPs, convergence also to a primal optimum as well as a linear convergence rate can be shown [Yang and Han 2016; Han 2022]. If problem (2.1) is a strictly convex QP, then ADMM converges q-linearly in the vector $w \doteq (z, \gamma/\rho)$, cf. Appendix A.5. Linear convergence for non-strictly convex QPs can also be shown, albeit to the solution set as the minimizer is not necessarily unique [Yang and Han 2016]. The decision variable y is feasible with respect to the subsystem constraints and the decision variable z satisfies the coupling constraints (2.1d) Feasibility of the consenus constraints (2.9d) is only achieved asymptotically. Termination criteria can be based for instance on the *dual residual* $s^{k+1} \doteq \rho(z^{k+1} - z^k)$ and the *primal residual* $r^{k+1} \doteq y^{k+1} - z^{k+1}$ [Boyd et al. 2011, Chapter 3].

Closed-loop stability, warm starting, and communication Thanks to the linear convergence rate, stability for linear-quadratic DMPC can be shown via RTI arguments [Zanelli 2021]. Chapter 4 will discuss this in greater detail. The method is warm-started with z_i^0 and γ_i^0 for all $i \in \mathcal{S}$, which can be taken from the previous MPC step. Choosing $\gamma_i^0 = E_i^\top \lambda^0$ for some $\lambda^0 \in \mathbb{R}^{n_c}$ is not strictly necessary, but ensures $M_{\text{avg}}\gamma^0 = 0$ and thus helps facilitate the averaging step as discussed in Lemma 2.1. Otherwise, the dual variables γ_i also need to be communicated between subsystems to evaluate the average via (2.14). This is the case if the subsystems execute ADMM asynchronously as only synchronous execution ensures $M_{\text{avg}}\gamma^{k+1} = 0$ via Lemma 2.1 iii) and iv).

Extensions and applications Since the inception of ADMM in the 1970s [Glowinski and Marroco 1975; Gabay and Mercier 1976], many extensions and subvariants have been explored such as *generalized ADMM* [Deng and Yin 2016] and *multi-block proximal ADMM* [Yashtini 2021]. Especially asynchronous variants seem promising in the context of DMPC as they aim to reduce the time spent waiting for neighbors [Burk et al. 2021c]. DMPC based on ADMM has been studied in the context of e.g. robot formations [Van Parys and Pipeleers 2017; Burk et al. 2021b] and energy systems [Liu et al. 2019; Behrunani et al. 2024]. Moreover, ADMM forms the basis for the open-source DMPC codes omg-tools [Van Parys and Pipeleers 2017] and GRAMPC-D [Burk et al. 2021a].

Application to nonlinear DMPC ADMM has been found to work well in practice for nonlinear DMPC in experiments with a four tank system [Hentzelt et al. 2014] and robot formations [Van Parys and Pipeleers 2017]. From a theoretical point of view, however, convergence guarantees for ADMM that would be meaningful for nonlinear DMPC appear to be unavailable because of the non-convex constraint sets $\mathbb{Z}_i$. In general, there exist convergence guarantees for ADMM also in the presence of non-convex constraints [Li and Pong 2015; Wang et al. 2019; Themelis and Patrinos 2020]. These discussions shift the non-convexity into the objective via *indicator functions* and consider more general two-block or even multi-block constraints than the consensus constraint (2.9d). However, the available convergence guarantees require assumptions on the smoothness of the objectives and the rank of the constraint matrices which are not satisfied by the decentralized ADMM

versions used in DMPC. For instance, the result in [Themelis and Patrinos 2020] would require to solve all OCPs with non-convex constraints in a central coordinator, which would contradict the idea of decomposing computationally expensive steps in DMPC among subsystems. Nonetheless, ADMM has the potential to be used in nonlinear DMPC and respective stability guarantees have been derived in [Bestler and Graichen 2019] under the assumption that ADMM converges linearly to the OCP minimizer. One disadvantage in practical implementations can be that Step 3 requires each subsystem to solve a non-convex NLP, which was found to be the computational bottleneck in experiments with embedded hardware [Van Parys and Pipeleers 2017].

2.2.3 NLP Formulation for ADMM with one Communication Round per Iteration

As discussed in Remark 2.1, the above ADMM variant decentralizes the update for z^{k+1} via an averaging step which requires two rounds of communication: Step 1 exchanges w between neighbors and Step 3 exchanges the averaged values $\bar{x}$. It is, however, also possible to reformulate the partially separable NLP such that the above ADMM variant requires only one round of communication per iteration. This can be of interest in decentralized implementations if the delay associated with two communication rounds is found to be a bottleneck.

Recall that we reformulate the cooperative OCP by introducing state copies w_{ji} for all $j \in \mathcal{N}_i^{\text{in}}$ and for all $i \in \mathcal{S}$ as demonstrated in Example 2.1. The following example illustrates how this reformulation affects the ADMM averaging matrix M_{avg}.

Example 2.2 (OCP reformulation for ADMM with two communication rounds per iteration)**.** *Consider three subsystems with states x_1, x_2, and x_3 and suppose the first and last subsystems optimize over equality constraints $g_1(x_1, x_2) = 0$ and $g_3(x_3, x_2) = 0$, respectively. For simplicity, we do not specify further components of the OCP and assume that the prediction horizon is $N = 1$. The procedure outlined above for formulating a partially separable program thus introduces the copies w_{21} and w_{23} of state x_2 and assigns those copies as decision variables to the first and last subsystem, i.e., $z_1 \doteq (x_1, w_{21})$, $z_2 \doteq x_2$, and $z_3 = (x_3, w_{23})$ such that the subsystem equality constraints become $g_1(x_1, w_{21}) = 0$ and $g_3(x_3, w_{23}) = 0$. The coupling constraints* (2.1d) *read*

$$\underbrace{\begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix}}_{E_1} z_1 + \underbrace{\begin{bmatrix} 1 \\ 1 \end{bmatrix}}_{E_2} z_2 + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix}}_{E_3} z_3 = 0.$$

The averaging matrix thus is given by

$$M_{\text{avg}} = (I - E^\top (EE^\top)^{-1} E) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1/3 & 1/3 & 0 & 1/3 \\ 0 & 1/3 & 1/3 & 0 & 1/3 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1/3 & 1/3 & 0 & 1/3 \end{bmatrix}$$

and the averaging update for state x_2 becomes

$$\bar{x}_2^{k+1} = \bar{w}_{21}^{k+1} = \bar{w}_{23}^{k+1} = \frac{1}{3}\left(x_2^k + w_{21}^k + w_{32}^k\right).$$

The procedure outlined in Remark 2.2 evaluates this computation by having subsystems one and three send w_{21}^k and w_{23}^k to subsystem two, having subsystem two then evaluate the above average to obtain $\bar{x}_2^{k+1}$, and by then sending $\bar{x}_2^{k+1}$ to subsystems one and three. □

As an alternative to the above reformulation, it is also possible to not only introduce copies w_{ij} of state x_i, but additional copies v_{ij} which then become decision variables of subsystem $i \in \mathcal{S}$. As before, we assign the copies w_{ij} as decision variables to the out-neighbors $j \in \mathcal{N}_i^{\text{out}}$ for evaluating expressions that contain the state x_i. The copies v_{ij} become additional decision variables of subsystem i, i.e. subsystem i optimizes over additional copies of its own state x_i. This allows to formulate the coupling constraints (2.1d) as $v_{ij} - w_{ij} = 0$ instead of $x_i - w_{ij} = 0$, which leads to an ADMM variant with only one communication round per iteration, as we show below. The price to pay, however, is that the OCP then contains additional decision variables and that subsystem i is assigned additional subsystem equality constraints $x_i - v_{ij} = 0$ for all $j \in \mathcal{N}_i^{\text{out}}$. That is, consistency between the original state x_j and copied variables is not enforced via the coupling constraint (2.1d), but via additional equality constraints (2.1b).

Example 2.3 (OCP reformulation for ADMM with one communication round per iteration)**.** *Consider again the three subsystems of Example 2.2 with states x_1, x_2, and x_3 and equality constraints $g_1(x_1, x_2) = 0$ and $g_3(x_3, x_2) = 0$. We introduce the copies w_{21}, w_{23}, v_{21}, and v_{23} of state x_2 and define the decision variables $z_1 \doteq (x_1, w_{21})$, $z_2 \doteq (x_2, v_{21}, v_{23})$, and $z_3 = (x_3, w_{23})$. The subsystem equality constraints for the partially separable NLP then read $g_1(x_1, w_{21}) = 0$, $g_2 \doteq (x_2 - v_{21}, x_2 - v_{23}) = (0, 0)$, and $g_3 = (x_3, w_{23}) = 0$. The coupling constraints become*

$$\begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix} z_1 + \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} z_2 + \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix} z_3 = 0$$

and the ADMM averaging matrix reads

$$M_{\text{avg}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0.5 \end{bmatrix}.$$

The key difference compared to the reformulation of Example 2.2 is that each row of M_{avg} now contains at most two non-zero elements. Put differently, each average requires information of at most two subsystems,

$$\bar{w}_{21}^{k+1} = \bar{v}_{21}^{k+1} = \frac{1}{2}\left(w_{21}^k + v_{21}^k\right), \quad \bar{w}_{23}^{k+1} = \bar{v}_{23}^{k+1} = \frac{1}{2}\left(w_{23}^k + v_{23}^k\right).$$

Thus, neighbors can exchange both w and v simultaneously and let each of the two coupled subsystems evaluate the average. □

This alternative reformulates the OCP as a 2-assigned partially separable program [Engelmann et al. 2020] where each column of the centralized coupling matrix E has at most one non-zero entry. Let $M_{\text{avg}}\gamma^k = 0$. The procedure for decentralizing the z-update in ADMM then becomes:

1. For all $i \in \mathcal{S}$, send $\boldsymbol{w}_{ji}^{k+1}$ to and receive $\boldsymbol{v}_{ji}^{k+1}$ from all $j \in \mathcal{N}_i^{\text{in}}$; send $\boldsymbol{v}_{ij}^{k+1}$ to and receive $\boldsymbol{w}_{ij}^{k+1}$ from all $j \in \mathcal{N}_i^{\text{out}}$.

2. For all $i \in \mathcal{S}$, compute the averages

$$\bar{\boldsymbol{v}}_{ij}^{k+1} = \frac{1}{2}\left(\boldsymbol{v}_{ij}^{k+1} + \boldsymbol{w}_{ij}^{k+1}\right) \quad \forall j \in \mathcal{N}_i^{\text{out}} \quad \text{and}$$
$$\bar{\boldsymbol{w}}_{ji}^{k+1} = \frac{1}{2}\left(\boldsymbol{v}_{ji}^{k+1} + \boldsymbol{w}_{ji}^{k+1}\right) \quad \forall j \in \mathcal{N}_i^{\text{in}}.$$

Notice that this reformulation does not change the number of variables that must be communicated between neighbors compared to the reformulation of Example 2.1. Rather, the difference is that instead of communicating $\bar{x}^{k+1}$ in Step 3 of Remark 2.1, subsystems here communicate v^{k+1} in Step 1 concurrently to w^{k+1} before forming the average. A potential advantage in applications thus is that all communication of ADMM can happen in one communication round.

To summarize, this subsection has presented an alternative for reformulating OCPs as partially separable programs such that ADMM only requires one communication round per iteration. The disadvantage, however, is that this requires additional decision variables and subsystem equality constraints. Notably, the reformulation discussed in this section only affects the components of the partially separable NLP (2.1) whereas the ADMM iterations continue to read as presented in Algorithm 2.3. The theoretical analysis of ADMM applies to partially separable NLPs and is thus valid for both types of OCP reformulations, with or without copies v_{ij}. What changes, however, is the size of the subsystem programs to be solved in Step 3 of Algorithm 2.3, the number of communication rounds for Step 4, and the length of the Lagrange multiplier vector γ. Thus, the theoretical analyses to be developed in the remainder of this thesis apply to both OCP reformulations. We note, however, that all numerical and experimental results presented in this work rely on the reformulation without v_{ij} and thus require two communication rounds per ADMM iteration.

2.2.4 Essentially Decentralized Active Set Method

We next recall the bi-level d-ASM method from [Stomberg et al. 2021] and which is summarized in Algorithm 2.4, combining a primal active set method on the outer level [Nocedal and Wright 2006, Chapter 16] with essentially decentralized Conjugate Gradients (d-CG) on the inner level [Engelmann et al. 2020]. To this end, we define the active set of subsystem $i \in \mathcal{S}$ as

$$\mathcal{A}_i(z_i) \doteq \left\{ j \in \{1, \dots, n_{h,i}\} \,\middle|\, \left[A_i^{\text{I}}\right]_j z_i = \left[b_i^{\text{I}}\right]_j \right\}.$$

Algorithm 2.4 Essentially decentralized ASM [Stomberg et al. 2021].

Initialization: $k = 0$, $z^0 \in \mathbb{Z} \cap \mathbb{E}$, $\mathcal{W}_i^0$ for all $i \in \mathcal{S}$, ε
while true **do**
  Compute f_i^k, C_i^k, Z_i^k, and Y_i^k for all $i \in \mathcal{S}$.
  Condense QP (2.16) by computing $\bar{H}_i^k$, $\bar{f}_i^k$, $\bar{E}_i^k$, S_i^k, and s_i^k for all $i \in \mathcal{S}$.
  Solve the linear system of equations (2.23) with d-CG and obtain λ^k.
  Obtain step directions Δz_i^k via (2.24) for all $i \in \mathcal{S}$.
  if $\|\Delta z_i^k\|_\infty < \varepsilon$ for all $i \in \mathcal{S}$ **then**
    Compute Lagrange multipliers μ_i^k via (2.20) for all $i \in \mathcal{S}$.
    if $[\mu_i^{\mathcal{I}k}]_j \geq 0$ for all $j \in \mathcal{W}_i^k$ and for all $i \in \mathcal{S}$ **then**
      return z_i^k for all $i \in \mathcal{S}$.
    else
      Find the constraint and subsystem $(i^\star, j^\star) = \operatorname{argmin}[\mu_i^{\mathrm{I}k}]_j$ s.t. , $j \in \mathcal{W}_i^k$.
      Remove constraint $j^\star$ from the working set by updating $\mathcal{W}_{i^\star}^{k+1} = \mathcal{W}_{i^\star}^k \setminus \{j^\star\}$.
      Set $\mathcal{W}_i^{k+1} = W_i^k$ for all $i \in \mathcal{S} \setminus \{i^\star\}$.
      $k \leftarrow k + 1$
      go to Step 3.
    end if
  else
    Compute a feasible step length α_i^k via (2.17) for all $i \in \mathcal{S}$.
    Select α^k and the blocking subsystem $i^\star$ and constraint $j^\star$ via (2.18)–(2.19).
    Add the blocking constraint to the working set $\mathcal{W}_{i^\star}^{k+1} = \mathcal{W}_{i^\star}^{k+1} \cup \{j^\star\}$.
    Set $\mathcal{W}_i^{k+1} = \mathcal{W}_i^k$ for all $i \in \mathcal{S} \setminus \{i^\star\}$.
    $z_i^{k+1} = z_i^k + \alpha^k \Delta z_i^k$ for all $i \in \mathcal{S}$.
    $k \leftarrow k + 1$
    go to Step 3.
  end if
end while

Starting from a feasible point $z^k \in \mathbb{Z} \cap \mathbb{E}$, each subsystem treats a subset of inequality constraints, referred to as the working set $\mathcal{W}_i^k \subseteq \mathcal{A}_i(z_i^k)$, as equality constraints while disregarding the remaining inequality constraints. The d-ASM update reads

$$z_i^{k+1} = z_i^k + \alpha^k \Delta z_i^k \quad \forall i \in \mathcal{S},$$

where the step size $\alpha^k \in [0, 1]$ is the same for all subsystems. The step directions $\Delta z_i^k \in \mathbb{R}^{n_i}$ are found by solving the equality-constrained QP

$$\left(\Delta z_1^k, \ldots, \Delta z_S^k\right) = \underset{\Delta z_1, \ldots, \Delta z_S}{\operatorname{argmin}} \quad \sum_{i \in \mathcal{S}} \left(\frac{1}{2} \Delta z_i^\top H_i \Delta z_i + f_i^{k\top} \Delta z_i \right) \tag{2.16a}$$

$$\text{subject to} \quad C_i^k \Delta z_i = d_i^k \mid \mu_i^k \quad \forall i \in \mathcal{S} \tag{2.16b}$$

$$\sum_{i \in \mathcal{S}} E_i \Delta z_i = b^k \mid \lambda^k, \tag{2.16c}$$

where

$$f_i^k \doteq H_i z_i^k + a_i, \quad C_i^k \doteq \begin{bmatrix} A_i^{\mathrm{E}} \\ \left[A_i^{\mathrm{I}}\right]_{\mathcal{W}_i^k} \end{bmatrix}, \quad d_i^k \doteq \begin{bmatrix} b_i^{\mathrm{E}} - A_i^{\mathrm{E}} z_i^k \\ \left[b_i^{\mathrm{I}}\right]_{\mathcal{W}_i^k} - \left[A_i^{\mathrm{I}}\right]_{\mathcal{W}_i^k} z_i^k \end{bmatrix}, \quad \text{and} \quad b^k \doteq b - \sum_{i \in \mathcal{S}} E_i z_i^k.$$

The above notation indicates that the matrix $[A_i^{\mathrm{I}}]_{\mathcal{W}_i^k}$ is the horizontal concatenation of the rows $[A_i^{\mathrm{j}}]_j$ for all $j \in \mathcal{W}_i^k$. Note that the right-hand sides d_i^k and b^k vanish for all $i \in \mathcal{S}$, if z^k is feasible. However, we include d_i^k and b^k in our derivation, because they allow to find feasible initializations with essentially decentralized computation as described later. For now, we assume a feasible iterate z^k is available.

The step size α^k is determined such that primal feasibility with respect to all constraints is maintained. If $z^k \in \mathbb{Z}$, then the constraints (2.16b) imply $C_i^k \Delta z_i = d_i^k = 0$ and thus $C_i^k(z_i^k + \alpha^k \Delta z_i^k) = 0$ for any α^k, i.e., any step length preserves feasibility with respect to the subsystem equality constraints as well as the inequality constraints inside the working set. The same applies to the coupling constraints due to (2.16c). However, feasibility only holds for sufficiently small step lengths for inequality constraints $j \neq \mathcal{W}_i^k$ if $[A_i^{\mathcal{I}}]_j \Delta z_i^k > 0$, because $[A_i^{\mathcal{I}}]_j(z_i^k + \alpha^k \Delta z_i^k) > [b_i^{\mathcal{I}}]_j$ if α^k is too large. Thus, each subsystem checks for potential increases of inactive inequality constraints along the step direction Δz_i^k and computes a feasible step length [Nocedal and Wright 2006, Equation 16.41]

$$\alpha_i^k \doteq \min\left(1, \min_{j \notin \mathcal{W}_i^k, [A_i^{\mathrm{I}}]_j \Delta z_i^k > 0} \frac{[b_i^{\mathrm{I}}]_j - [A_i^{\mathrm{I}}]_j z_i^k}{[A_i^{\mathrm{I}}]_j \Delta z_i^k}\right). \tag{2.17}$$

All subsystems then apply the same step length

$$\alpha^k \doteq \min\left(\alpha_1^k, \ldots, \alpha_S^k\right) \tag{2.18}$$

to preserve feasibility. If $\alpha^k < 1$, then the blocking subsystem and constraint are determined by

$$(i^\star, j^\star) = \operatorname*{argmin}_{i \in \mathcal{S}, j \notin \mathcal{W}_i^k, [A_i^{\mathrm{I}}]_j \Delta z_i^k > 0} \frac{[b_i^{\mathrm{I}}]_j - [A_i^{\mathrm{I}}]_j z_i^k}{[A_i^{\mathrm{I}}]_j \Delta z_i^k} \tag{2.19}$$

and the constraint $j^\star$ is added to the working set of subsystem $i^\star$. Then, all subsystems update z_i and form a new equality-constrained QP (2.16). This process is repeated until $\Delta z_i^k = 0$ for all $i \in \mathcal{S}$, which indicates that an optimum over the active constraints has been reached. Then, the subsystem constraint Lagrange multipliers μ_i^k are computed through the KKT system of QP (2.16),

$$\mu_i^k = \left(C_i^k C_i^{k\top}\right)^{-1} C_i^k\Big(- H_i \underbrace{\Delta z_i^k}_{0} - f_i^k - E_i^\top \lambda^k\Big) = \left(C_i^k C_i^{k\top}\right)^{-1} C_i^k\Big(- f_i^k - E_i^\top \lambda^k\Big) \quad \forall i \in \mathcal{S}. \tag{2.20}$$

If $[\mu_i^k]_j \geq 0$ for all $j \in \mathcal{W}_i^k$, then dual feasibility and thus a solution to QP (2.2) has been reached. Otherwise, a negative Lagrange multiplier to an inequality constraint implies that a lower objective

value can be found by removing the respective constraint from the working set. We remove the constraint corresponding to the most negative Lagrange multiplier from the working set and then construct a new equality-constrained QP (2.16) [Nocedal and Wright 2006, Chapter 16].

Inspired by *Bi-Level ALADIN* (BL-ALADIN), d-ASM solves QP (2.16) with essentially decentralized Conjugate Gradients (d-CG) [Engelmann et al. 2020]. To this end, we recast QP (2.16) as a linear system of equations such that d-CG becomes applicable. Crucially, we condense QP (2.16) from n decision variables to a system of equations in n_c variables. This reduces the number of d-CG iterations and lowers the communication footprint. To condense the QP, we apply the null-space method to the subsystem constraints (2.16b) [Nocedal and Wright 2006, Chapter 16]. Let

$$\Delta z_i^k = Z_i^k v_i^k + Y_i^k w_i^k, \tag{2.21}$$

where the columns of $Z_i^k \in \mathbb{R}^{n_i \times (n_i - n_{g_i}^k)}$ form a basis of the null space of $C_i^k \in \mathbb{R}^{n_{g_i}^k \times n_i}$ and where $Y_i^k \in \mathbb{R}^{n_i \times n_{g_i}^k}$ is such that the square matrix $[Y_i^k, Z_i^k]$ has full rank for all $i \in \mathcal{S}$. For instance, Y_i^k can be chosen such that the columns of Y_i^k form a basis for the range space of $C_i^{k\top}$ for all $i \in \mathcal{S}$. We define the centralized constraint Jacobian $C^k \doteq \operatorname{diag}(C_1^k, \dots, C_S^k)$ and the centralized null-space matrix $Z^k \doteq \operatorname{diag}(Z_1^k, \dots, Z_S^k)$.

Assumption 2.2 (Assumptions for d-ASM)**.** *The following holds for all $i \in \mathcal{S}$ and for all $k \geq 0$:*

i) The Jacobian C_i^k has full row rank.

ii) The condensed Hessian $\bar{H}_i^k \doteq Z_i^{k\top} H_i Z_i^k$ is positive definite.

iii) The matrix $[C^{k\top} E^\top]^\top$ has full row rank, i.e. LICQ holds. □

We note that Assumption 2.2 i) is weaker than Assumption 2.2 iii) and is only specified separately to pinpoint where LICQ is required. Assumptions 2.2 i) and ii) ensure that the matrices $C_i^k Y_i^k$ and $\bar{H}_i^k$ are regular, which we need for condensing the QP via the null-space method. In a numerical implementation, the full row rank of C_i^k only needs to be checked at the first iteration. This is because full row rank of C_i^0 implies full row rank of C_i^k for all $k > 0$ by design of d-ASM [Nocedal and Wright 2006, p. 476]. The assumption on the positive definiteness of $\bar{H}_i$ is mild and holds if H_i is positive definite, because Z_i^k has full column rank.

Inserting the null-space approach (2.21) into (2.16) yields the condensed QP

$$\min_{v_1,\dots,v_S} \sum_{i\in\mathcal{S}} \frac{1}{2} v_i^\top \bar{H}_i^k v_i + \bar{f}_i^{k\top} v_i \tag{2.22a}$$

$$\text{subject to} \quad \sum_{i\in\mathcal{S}} \left(\bar{E}_i^k v_i + E_i Y_i^k w_i^k \right) = b^k \mid \lambda^k, \tag{2.22b}$$

where

$$w_i^k \doteq (C_i^k Y_i^k)^{-1} d_i^k, \quad \bar{f}_i^k \doteq Z_i^{k\top} f_i^k + Z_i^{k\top} H_i Y_i^k w_i^k, \quad \text{and} \quad \bar{E}_i^k \doteq E_i Z_i^k.$$

To obtain a linear system of equations for d-CG, we apply the Schur-complement method to the condensed QP [Nocedal and Wright 2006, Chapter 16]. Solving the KKT conditions

$$\begin{bmatrix} \bar{H}_1^k & \dots & 0 & \bar{E}_i^{k\top} \\ 0 & \ddots & 0 & \vdots \\ 0 & \dots & \bar{H}_S^k & \bar{E}_S^{k\top} \\ \bar{E}_1^k & \dots & \bar{E}_S^k & 0 \end{bmatrix} \begin{bmatrix} v_1^k \\ \vdots \\ v_S^k \\ \lambda \end{bmatrix} = \begin{bmatrix} -\bar{f}_1^k \\ \vdots \\ -\bar{f}_S^k \\ b^k - \sum_{i\in\mathcal{S}} E_i Y_i^k w_i^k \end{bmatrix}$$

for v_i^k yields $v_i^k = \bar{H}_i^{-1}\left(-\bar{f}_i^k - \bar{E}_i^{k\top}\lambda^k\right)$ for all $i \in \mathcal{S}$. Inserting back into the bottom block rows of the KKT conditions yields the linear system of equations

$$\Big(\sum_{i\in\mathcal{S}} \underbrace{\bar{E}_i^k (\bar{H}_i^k)^{-1} \bar{E}_i^{k\top}}_{\doteq S_i^k}\Big)\lambda^k = \sum_{i\in\mathcal{S}} \Big(\underbrace{E_i Y_i^k w_i^k - \bar{E}_i^k (\bar{H}_i^k)^{-1} \bar{f}_i - \frac{b^k}{S}}_{\doteq s_i^k}\Big). \tag{2.23}$$

The system (2.23) is solved via d-CG and backsubstitution yields

$$\Delta z_i^k = Z_i^k (\bar{H}_i^k)^{-1}\left(-\bar{f}_i^k - \bar{E}_i^{k\top}\lambda^k\right) + Y_i^k (C_i^k Y_i^k)^{-1} d_i^k \quad \forall i \in \mathcal{S}. \tag{2.24}$$

To solve (2.23) with d-CG, we have to ensure that the matrix $S^k \doteq \sum_{i\in\mathcal{S}} S_i^k$ is positive definite, cf. Appendix A.6. To this end, we derive the following lemma which has appeared in a slightly different form for bi-level ALADIN [Engelmann et al. 2020] and which we prove here for a self-contained presentation of d-ASM.

Lemma 2.2 (Applicability of d-CG to d-ASM)**.** *Let Assumptions 2.1 and 2.2 hold. Then the matrix $S^k = \sum_{i\in\mathcal{S}} S_i^k$ is symmetric positive definite for all $k \geq 0$.* □

Proof. We first prove, (a), that the matrix $\bar{E}^k \doteq [\bar{E}_1^k, \dots, \bar{E}_S^k] \in \mathbb{R}^{n_c \times (n - n_g^k)}$ has full row rank. Then, (b), we show that full row rank of $\bar{E}^k$ and positive definiteness of $\bar{H}^{-1}$ together imply positive definiteness of S.

(a) By Assumption 2.2 i), the matrix $C^k \in \mathbb{R}^{n_g^k \times n}$ has full row rank and thus the null-space basis $Z^k \in \mathbb{R}^{n \times (n - n_g^k)}$ has full column rank. Furthermore, we have $C^k Z^k = 0$ by design of Z^k. The matrix $[C^{k\top} E^\top]^\top \in \mathbb{R}^{(n_g^k + n_c) \times n}$ has full row rank because of LICQ. By Sylvester's inequality [Horn and Johnson 2013, Section 0.4.5], we have

$$\operatorname{rank}\left(\begin{bmatrix} C^k \\ E \end{bmatrix} Z^k\right) \geq \operatorname{rank}\left(\begin{bmatrix} C^k \\ E \end{bmatrix}\right) + \operatorname{rank}\left(Z^k\right) - n = (n_g^k + n_c) + (n - n_g^k) - n = n_c.$$

Since

$$\begin{bmatrix} C^k \\ E \end{bmatrix} Z^k = \begin{bmatrix} 0 \\ \bar{E}^k \end{bmatrix},$$

we have $\operatorname{rank}\left(\bar{E}^k\right) = n_c$.

(b) We have to show that $\lambda^\top \bar{E}^k \left(\bar{H}^k\right)^{-1} \bar{E}^{k\top} \lambda > 0$ for all $\lambda \neq 0$. Let $v = \bar{E}^{k\top}\lambda$. By (a), $\bar{E}^k$ has full row rank and thus $\bar{E}^{k\top}$ has full column rank. The rank-nullity theorem [Horn and Johnson 2013, Equation 0.2.3.1] thus implies that the dimension of the null space of $\bar{E}^{k\top}$ is zero and hence $v = 0 \iff \lambda = 0$. Assumption 2.2 ii) states that $\left(\bar{H}_i^k\right)^{-1}$ is positive definite. We therefore have

$$\lambda^\top \bar{E}^k \left(\bar{H}^k\right)^{-1} \bar{E}^{k\top} \lambda = v^\top \left(\bar{H}^k\right)^{-1} v > 0 \quad \forall \lambda \neq 0.$$

Symmetry of S^k follows from the definition of the matrices S_i^k. This concludes the proof. ■

Computing a feasible initialization Algorithm 2.4 requires a feasible initialization $z^0 \in \mathbb{Z} \cap \mathbb{E}$ to maintain feasibility for any step length α^k. In general, obtaining an initial guess that satisfies both the subsystem and coupling constraints by distributed or decentralized computation can be challenging. However, we can initalize d-ASM with essentially decentralized computation as follows. The constraints (2.16b) and (2.16c) ensure that all subsystem equality constraints, inequality constraints in the working sets, and coupling constraints will be satisfied by $z_i^{k+1} = z_i^k + \Delta z_i^k$ for all $i \in \mathcal{S}$. That is, if we apply the step length $\alpha^k = 1$, then z^{k+1} is guaranteed to satisfy all constraints except for inequality constraints outside the working sets, even if z^k is infeasible. We can thus warm-start d-ASM by initializing with an empty working set, solving QP (2.16) for Δz_i^k for all $i \in \mathcal{S}$, finding a guess by $z_i^{k+1} = z_i^k + \Delta z_i^k$, and by adding violated inequality constraints to the working sets. This process is repeated until a feasible initialization has been found. We note that this heuristic approach is not guaranteed to find feasible initializations for all QPs, but is guaranteed to work for OCPs with input box constraints and no state constraints.

Assumptions, convergence, and feasibility Linear independence of the active subsystem constraints and strict convexity allow to reformulate the equality-constrained QP as a linear system of equations via the null-space and Schur-complement methods. LICQ, i.e., linear independence of the active subsystem *and* coupling constraints, further guarantees the well-definedness and finite-time convergence of d-CG. If no cycling occurs, then the method converges in finitely many outer iterations [Nocedal and Wright 2006, Chapter 16]. All outer iterates z^k are feasible with respect to the subsystem *and* coupling constraints.

Closed-loop stability, warm-starting, and communication The feasible-side convergence of d-ASM guarantees closed-loop stability for appropriate OCP designs [Scokaert et al. 1999]. The method can be warm-started with the working sets and solution of the previous MPC step. However, a new feasible initialization is required in the presence of model-plant mismatch or other disturbances. Algorithm 2.4 requires global communication for finding a feasible step length α^k in Line 19 and for determining the smallest Lagrange multiplier in Line 12. Furthermore, d-CG requires to communicate $2n_c$ floats on a neighbor-to-neighbor basis and two floats globally per d-CG iteration, cf. Appendix A.6.

2.2.5 Jacobi Iterations

Some of the first optimization algorithms that were explored for cooperative DMPC are based on parallel Jacobi iterations [Bertsekas and Tsitsiklis 1989, Chapter 3]. Early DMPC approaches using Jacobi iterations consider coupling in the linear system dynamics and decoupled input constraints [Venkat et al. 2005; Stewart et al. 2010]. This section summarizes an overlapping variant that is applicable OCPs with coupling in the cost, dynamics, and input and state constraints [Doan et al. 2017].

To present Jacobi iterations, we deviate from the notation used in QP (2.2) and the other algorithms presented in this section. Instead, we write the centralized OCP as

$$\min_{z} \frac{1}{2} z^\top H z + g^\top z \tag{2.25a}$$

$$\text{subject to} \quad C^{\mathrm{E}} z = b^{\mathrm{E}}, \tag{2.25b}$$

$$C^{\mathrm{I}} z \leq b^{\mathrm{I}}, \tag{2.25c}$$

with the centralized decision variable $z \doteq (\boldsymbol{x}, \boldsymbol{u}) \in \mathbb{R}^n$. That is, we do not introduce trajectory copies to couple the subsystems via (2.2d), but instead couple neighboring systems through the matrices H, C^{E}, and C^{I}. Each subsystem minimzes QP (2.25) over a subset of the components of z. The implementation and performance of the algorithm depends on the assignment of decision variables to subsystems. One option is to choose the overlapping decomposition [Doan et al. 2017]

$$z_i \doteq (\boldsymbol{x}_j, \boldsymbol{u}_j)_{j \in \{\mathcal{N}_i \cup \{i\}\}} \in \mathbb{R}^{n_i}, \tag{2.26}$$

where subsysem i is assigned its own and neighboring trajectories as decision variables. Let $z_{-i} \doteq (\boldsymbol{x}_j, \boldsymbol{u}_j)_{j \notin \{\mathcal{N}_i \cup \{i\}\}} \in \mathbb{R}^{n - n_i}$. That is, the vectors z_i and z_{-i} together contain all centralized decision variables z. We rewrite the centralized objective as $\phi : \mathbb{R}^{n_i} \times \mathbb{R}^{n-n_i} \to \mathbb{R}$, $\phi(z_i, z_{-i}) \doteq z^\top H z / 2 + g^\top z$. At iteration k, and given an iterate z_{-i}^k, subsystem i solves

$$z_i^{k,\star} \doteq \operatorname*{argmin}_{z_i} \phi(z_i, z_{-i}^k) \tag{2.27a}$$

$$\text{subject to} \quad C_i^{\mathrm{E}} z_i + C_{-i}^{\mathrm{E}} z_{-i}^k = b_i^{\mathrm{E}}, \tag{2.27b}$$

$$C_i^{\mathrm{I}} z_i + C_{-i}^{\mathrm{I}} z_{-i}^k \leq b_i^{\mathrm{I}} \tag{2.27c}$$

to obtain the intermediate iterate $z_i^{k,\star} \in \mathbb{R}^{n_i}$. The notation in (2.27a) denotes that subsystem i minimizes the centralized objective with respect to z_i while keeping z_{-i}^k constant. The constraints (2.27b) and (2.27c) respectively include the non-zero rows of (2.25b) and (2.25c) for the components of z_i. For the chosen overlapping decomposition, this includes the system dynamics and state and input constraints of subsystem i and its neighbors. Subsystem i then combines the optimized variables $z_i^{k,\star}$ with z_{-i}^k to form a guess $z^{k,i} \in \mathbb{R}^n$ for the centralized decision variables. The subsystems then communicate and compute the next centralized iterate in Step 5 of Algorithm 2.5 as a convex combination of the vectors $z^{k,i}$ with weights

$$\omega_i \geq 0 \quad \forall i \in \mathcal{S} \quad \text{and} \quad \sum_{i \in \mathcal{S}} \omega_i = 1. \tag{2.28}$$

Decentralization Algorithm 2.5 is decentralized, because the sparsity of H, C_i^{E}, and C_i^{I} allows to decompose the summation in Step 5 as follows. Consider the convex combination of the predicted state and input trajectories of subsystem i,

$$\begin{bmatrix} \boldsymbol{x}_i^{k+1} \\ \boldsymbol{u}_i^{k+1} \end{bmatrix} = \omega_i \begin{bmatrix} \boldsymbol{x}_i^{k,i} \\ \boldsymbol{u}_i^{k,i} \end{bmatrix} + \sum_{j \in \mathcal{N}_i} \omega_j \begin{bmatrix} \boldsymbol{x}_i^{k,j} \\ \boldsymbol{u}_i^{k,j} \end{bmatrix} + \sum_{p \in \mathcal{S} \setminus \{\mathcal{N}_i \cup \{i\}\}} \omega_p \begin{bmatrix} \boldsymbol{x}_i^{k,p} \\ \boldsymbol{u}_i^{k,p} \end{bmatrix},$$

where $\boldsymbol{x}_i^{k,j}$ and $\boldsymbol{u}_i^{k,j}$ refer to those components of $z^{k,j}$, $j \in \mathcal{S}$, that correspond to the predicted state and input trajectory of subsystem $i \in \mathcal{S}$, respectively. By definition (2.26) and by the design of the subsystem QPs (2.27), the subsystems $p \in \mathcal{S} \setminus \{\mathcal{N}_i \cup \{i\}\}$ do not minimize over $(\boldsymbol{x}_i, \boldsymbol{u}_i)$ and thus $(\boldsymbol{x}_i^{k,p}, \boldsymbol{u}_i^{k,p}) = (\boldsymbol{x}_i^k, \boldsymbol{u}_i^k)$. Therefore, subsystem i only needs to receive $(\boldsymbol{x}_i^{k,j}, \boldsymbol{u}_i^{k,j})$ from all $j \in \mathcal{N}_i$ and can evaluate

$$\begin{bmatrix} \boldsymbol{x}_i^{k+1} \\ \boldsymbol{u}_i^{k+1} \end{bmatrix} = \omega_i \begin{bmatrix} \boldsymbol{x}_i^{k,i} \\ \boldsymbol{u}_i^{k,i} \end{bmatrix} + \sum_{j \in \mathcal{N}_i} \omega_j \begin{bmatrix} \boldsymbol{x}_i^{k,j} \\ \boldsymbol{u}_i^{k,j} \end{bmatrix} + \sum_{p \in \mathcal{S} \setminus \{\mathcal{N}_i \cup \{i\}\}} \omega_p \begin{bmatrix} \boldsymbol{x}_i^{k} \\ \boldsymbol{u}_i^{k} \end{bmatrix},$$

where $(\boldsymbol{x}_i^k, \boldsymbol{u}_i^k)$ is available from the previous Jacobi iteration. The updated trajectories $(\boldsymbol{x}_i^{k+1}, \boldsymbol{u}_i^{k+1})$ are then sent to all subsystems $j \in \bar{\mathcal{N}}_i$, where the extended neighborhood $\bar{\mathcal{N}}_i \subseteq \mathcal{S}$ includes all subsystems which require the trajectories of subsystem i to form their parameter vector z_{-j} [Doan et al. 2017, Remark 3.2].

Assumptions, convergence, and termination If QP (2.25) is convex, then the objective $f(z^k) \doteq z^{k\top} H z^k/2 + g^\top z^k$ converges to a constant $c \geq 0$ [Doan et al. 2017, Lemma 3.7]. Convergence to a minimum is not guaranteed in general, but can be derived for chain-linked systems [Doan et al. 2017, Theorem 4.1]. The algorithm can be terminated if the step size $\|z^{k+1} - z^k\|$ is small [Venkat et al. 2005].

Feasibility and stability All iterates z_i^k are feasible with respect to the centralized constraints (2.25b) and (2.25c), because the constraints are convex, the iterates $z_i^{k,\star}$ are feasible, and because Step 5 of Algorithm 2.5 is a convex combination [Doan et al. 2017, Lemma 3.6]. Stability therefore follows for stabilizing OCP designs with appropriately designed terminal costs and constraints [Scokaert et al. 1999].

Warm starting and communication All initial iterates z_i^0 must be feasible, which complicates the decentralized implementation for problems with coupled constraints. The solution from the previous MPC step can be used, if there is no plant-model mismatch and if feasible predicted trajectories can be easily generated from the prior solution. While the method is decentralized, communication with the extended neighborhood defined above is required to distribute z^{k+1}. This is a difference compared to the methods previously described in this section, where neighbor-to-neighbor communication is limited to the neighborhood defined by the OCP coupling graph $\mathcal{G}$.

Algorithm 2.5 Jacobi iterations [Doan et al. 2017].

1: Initialization: $k = 0$, z^0 feasible, $\{\omega_i\}_{i\in\mathcal{S}}$ satisfying (2.28), $k_{\max}$
2: **while** $k < k_{\max}$ **do**
3: $\quad z_i^{k,\star} = \underset{z_i}{\operatorname{argmin}}\, \phi(z_i, z_{-i}^k) \quad$ s.t. $\quad C_i^{\mathrm{E}} z_i + C_{-i}^{\mathrm{E}} z_{-i}^k = b_i^{\mathrm{E}},\ C_i^{\mathrm{I}} z_i + C_{-i}^{\mathrm{I}} z_{-i}^k \leq b_i^{\mathrm{I}}$ for all $i \in \mathcal{S}$.
4: $\quad$ Merge $z_i^{k,\star}$ and z_{-i}^k into $z^{k,i}$ for all $i \in \mathcal{S}$.
5: $\quad z^{k+1} = \sum_{i\in\mathcal{S}} \omega_i z^{k,i}$.
6: $\quad$ Distribute z^{k+1} among the subsystems and form z_{-i}^{k+1} for all $i \in \mathcal{S}$.
7: $\quad k \leftarrow k + 1$
8: **end while**

2.3 Algorithms for Nonlinear DMPC

Recall the partially separable NLP (2.1) to be solved in each NMPC step,

$$\min_{z} \sum_{i\in\mathcal{S}} f_i(z_i) \tag{2.29a}$$

$$\text{subject to} \quad g_i(z_i) = 0 \mid \nu_i \quad \forall i \in \mathcal{S}, \tag{2.29b}$$

$$h_i(z_i) \leq 0 \mid \mu_i \quad \forall i \in \mathcal{S}, \tag{2.29c}$$

$$\sum_{i\in\mathcal{S}} E_i z_i = b \mid \lambda. \tag{2.29d}$$

2.3.1 Essentially Decentralized Interior Point Method

We next summarize the bi-level d-IP method for solving NLP (2.29) presented in [Engelmann et al. 2021b], which combines a centralized primal-dual interior point method on the outer level with with d-CG on the inner level. The interior point method on the outer level guarantees local convergence and d-CG on the inner level decomposes the computations among the subsystems. We reformulate the inequality constraints (2.29c) and form the barrier problem

$$\min_{z,w} \sum_{i\in\mathcal{S}} \left(f_i(z_i) - \mathbb{1}^\top \delta \ln(w_i)\right) \tag{2.30a}$$

$$\text{subject to} \quad g_i(z_i) = 0 \mid \nu_i \quad \forall i \in \mathcal{S}, \tag{2.30b}$$

$$h_i(z_i) + w_i = 0 \mid \mu_i \quad \forall i \in \mathcal{S}, \tag{2.30c}$$

$$w_i \geq 0 \qquad \forall i \in \mathcal{S}, \tag{2.30d}$$

$$\sum_{i\in\mathcal{S}} E_i z_i = b \mid \lambda, \tag{2.30e}$$

where $w_i \in \mathbb{R}^{n_{h,i}}$ are slack variables for all $i \in \mathcal{S}$, $w \doteq (w_1, \ldots, w_S)$, $\delta > 0$ is the barrier parameter, $\mathbb{1}$ is the column vector of appropriate dimension with all components equal to 1, and the $\ln(\cdot)$ is evaluated componentwise. In each outer iteration, d-IP applies one Newton step to (2.30) and

subsequently decreases δ. This process is repeated until a solution to NLP (2.29) is recovered in the limit $\delta \to 0$.

We define the primal-dual vector $p_i \doteq (z_i, w_i, \mu_i, \mu_i)$ and Lagrangian $L_i(p_i) \doteq f_i(z_i) - \mathbb{1}^\top \ln(w_i) + \nu_i^\top g_i(z_i) + \mu_i^\top (h_i(z_i) + w_i)$ for all $i \in \mathcal{S}$. Given a primal-dual iterate $(p_1^k, \dots, p_S^k, \lambda^k)$ and a barrier parameter δ^k, a Newton step for (2.30) reads

$$\begin{bmatrix} \nabla F_1^{k\top} & 0 & \dots & 0 & \bar{E}_1^\top \\ 0 & \nabla F_2^{k\top} & \dots & 0 & \bar{E}_2^\top \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & \nabla F_S^{k\top} & \bar{E}_S^\top \\ \bar{E}_1 & \bar{E}_2 & \dots & \bar{E}_S & 0 \end{bmatrix} \begin{bmatrix} \Delta p_1 \\ \Delta p_2 \\ \vdots \\ \Delta p_S \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} -F_1^k \\ -F_2^k \\ \vdots \\ -F_S^k \\ b - \sum_{i \in \mathcal{S}} E_i z_i^k \end{bmatrix}, \quad \text{where} \tag{2.31}$$

$$F_i^\delta(p_i, \lambda) \doteq \begin{bmatrix} \nabla f_i(z_i) + \nabla g_i(z_i)\nu_i + h_i(z_i)\mu_i + E_i^\top \lambda \\ -\delta^k W_i^{-1} \mathbb{1} + \mu_i \\ g_i(z_i) \\ h_i(z_i) + w_i \end{bmatrix} \quad \text{and}$$

$$\nabla F_i^{\delta\top}(p_i) = \begin{bmatrix} \nabla^2_{z_i z_i} L_i(p_i) & 0 & \nabla g_i(z_i) & \nabla h_i(z_i) \\ 0 & -W_i^{-1}\operatorname{diag}(\mu_i) & 0 & I \\ \nabla g_i(z_i)^\top & 0 & 0 & 0 \\ \nabla h_i(z_i)^\top & I & 0 & 0 \end{bmatrix}.$$

For all $i \in \mathcal{S}$, F_i^k and ∇F_i^k are shorthands for $F_i^{\delta^k}(p_i^k, \lambda^k)$ and $\nabla F_i^{\delta^k}(p_i^k, \lambda^k)$, respectively, $W_i \doteq \operatorname{diag}(w_i)$, and $\bar{E}_i \doteq \begin{bmatrix} E_i & 0 & 0 & 0 \end{bmatrix}$.

The key idea of d-IP is to decompose the Newton step among the subsystems via d-CG. Close to a regular KKT point, ∇F_i^k is invertible for all $i \in \mathcal{S}$ and we condense the Newton system (2.31) via the Schur complement method to obtain

$$\sum_{i \in \mathcal{S}} S_i^k \Delta\lambda = \sum_{i \in \mathcal{S}} s_i^k, \tag{2.32}$$

where

$$S_i^k \doteq \bar{E}_i \left(\nabla F_i^{k\top}\right)^{-1} \bar{E}_i^\top \quad \text{and} \quad s_i^k \doteq E_i z_i^k - \bar{E}_i^\top \left(\nabla F_i^{k\top}\right)^{-1} F_i^k - \frac{1}{|\mathcal{S}|} b. \tag{2.33}$$

Furthermore, the matrix $S \doteq \sum_{i \in \mathcal{S}} S_i$ is positive definite close to regular KKT points [Engelmann et al. 2021b] such that d-CG can be applied, cf. Appendix A.6. Crucially and in contrast to d-ASM, d-CG here is terminated early to reduce the number of inner iterations in d-IP. We define the residual to (2.33), $r_\lambda^k \doteq \sum_{i \in \mathcal{S}} S_i^k \Delta\lambda^k - \sum_{i \in \mathcal{S}} s_i^k$. The d-CG stopping criterion reads $\|r_\lambda^k\|_\infty \leq c_1 \left(\delta^k\right)^\eta$ with parameters $c_1 > 0$ and $\eta > 1$, which enforces more accurate d-CG solutions as $\delta^k \to 0$. Each subsystem obtains the Newton step direction $\Delta p_i^k = (\Delta z_i^k, \Delta w_i^k, \Delta \nu_i^k, \Delta \mu_i^k)$ via back substitution as given in Step 5 of Algorithm 2.6 and updates the barrier parameter

$$\delta_i^{k+1} = \theta \left(\frac{w_i^{k\top} \mu_i^k}{n_{h_i}} \right)^{\gamma+1} \tag{2.34}$$

Algorithm 2.6 d-IP [Engelmann et al. 2021b]

1: Initialization: $k = 0$, $\{p_i^0\}_{i\in\mathcal{S}}$, μ^0, λ^0,δ^0, ε
2: **while** $\|F^k\|_\infty > \varepsilon$ **do**
3: Local condensing: (S_i^k, s_i^k) via (2.33).
4: Solve $\sum_{i\in\mathcal{S}} S_i^k \Delta\lambda^k = \sum_{i\in\mathcal{S}} s_i^k$ with d-CG until $\|r_i^\lambda\|_\infty \leq c_1(\delta^k)^\eta$ for all $i \in \mathcal{S}$.
5: Compute the Newton step direction $\Delta p_i^k = -\left(\nabla F_i^{k\top}\right)^{-1}\left(F_i^k + \bar{E}_i^\top \Delta\lambda^k\right)$ for all $i \in \mathcal{S}$.
6: Compute δ_i^{k+1}, $\alpha_i^{\mathrm{p},k}$, and $\alpha_i^{\mathrm{d},k}$ for all $i \in \mathcal{S}$ via (2.34)–(2.35).
7: Compute $(\delta^{k+1}, \alpha^{\mathrm{p},k}, \alpha^{\mathrm{d},k})$ via (2.36).
8: Compute the next iterate $\lambda^{k+1} = \lambda^k + \alpha_\mathrm{d}^k \Delta\lambda^k$ and p_i^{k+1} via (2.37) for all $i \in \mathcal{S}$.
9: $k \leftarrow k + 1$
10: **end while**

with parameters $\gamma > 0$ and $\theta \approx 1$. The fraction-to-the-boundary-rule yields

$$\alpha_i^{\mathrm{p},k} = \min\left(\tau \min_{\Delta[w_i^k]_n<0}\left(-\frac{[w_i^k]_n}{\Delta[w_i^k]_n}\right), 1\right) \quad \text{and} \quad \alpha_i^{\mathrm{d},k} = \min\left(\tau \min_{\Delta[\mu_i^k]_n<0}\left(-\frac{[\mu_i^k]_n}{\Delta[\mu_i^k]_n}\right), 1\right), \tag{2.35}$$

where $\tau = 1 - (\delta^k)^\beta$ and $\beta > \gamma$ [Nocedal and Wright 2006, Chapter 19]. Then, the subsystems communicate and determine the centralized barrier parameter update and step sizes

$$\delta^{k+1} = \max_{i\in\mathcal{S}} \delta_i^{k+1}, \quad \alpha_\mathrm{p}^k \doteq \min_{i\in\mathcal{S}} \alpha_i^{\mathrm{p},k}, \quad \text{and} \quad \alpha_\mathrm{d}^k \doteq \min_{i\in\mathcal{S}} \alpha_i^{\mathrm{d},k}. \tag{2.36}$$

The centralized step sizes ensure $w_i^{k+1} > 0$ and $\mu_i^{k+1} > 0$ such that F_i^{k+1} and ∇F_i^{k+1} are well-defined for all $i \in \mathcal{S}$. Finally, each subsystem updates the primal and dual variables as

$$z_i^{k+1} = z_i^k + \alpha_\mathrm{p}\Delta z_i^k,\ w_i^{k+1} = w_i^k + \alpha_\mathrm{p}\Delta w_i^k,\ \nu_i^{k+1} = \nu_i^k + \alpha_\mathrm{d}\Delta\nu_i^k, \text{ and } \mu_i^{k+1} = \mu_i^k + \alpha_\mathrm{d}\Delta\mu_i^k. \tag{2.37}$$

Assumptions, initialization, feasibility, and convergence The method is well defined close to a regular KKT point [Engelmann et al. 2021b, Assumption 1]. The slack variable and Lagrange multiplier initialization must satisfy $w_i^0, \mu_i^0 > 0$ and the fraction-to-the-boundary rule (2.35) guarantees $w_i^k, \mu_i^k > 0$ for all $i \in \mathcal{S}$ and for all $l \in \mathbb{N}$. The iterates z_i^k are infeasible and satisfy the constraints (2.29b)–(2.29d) asymptotically. Local convergence of the primal-dual variables to a regular KKT point is guaranteed at a superlinear rate [Engelmann et al. 2021b, Theorem 2].

Communication and stability Algorithm 2.6 requires neighbor-to-neighbor communication inside d-CG as well as global communication for d-CG and to complete Step 7. These global communication steps only involve scalars and thus d-IP is essentially decentralized. To the best of our knowledge, there exist no stability guarantees for DMPC using d-IP yet, but the superlinear convergence potentially qualifies d-IP for analyses similar to [Graichen and Kugi 2010; Bestler and Graichen 2019; Zanelli et al. 2021].

Related work and applications A bi-level interior point method for convex NLPs with ADMM on the inner level is presented in [Pakazad et al. 2014]. Similar to d-IP, the algorithm requires centralized communication to determine step sizes on the outer level. A decentralized interior point method tailored to the DMPC of chain-linked linear systems is proposed in [Necoara et al. 2013], factorizing the Newton system with neighbor-to-neighbor communication. The method in [Necoara and Suykens 2009], also for convex problems, can be seen as a hybrid between the algorithm of [Necoara et al. 2013] and d-IP: The Newton system on the outer level is transformed into a linear system of equations for the coupling constraint Lagrange multiplier λ and a sparsity-exploiting factorization is used on the inner level. A distributed interior point method for non-convex NLPs is proposed in [Hult et al. 2022], where the coupled part of the Newton system is solved on a central coordinator. A distributed interior point method splitting a convex NLP into small-scale NLPs to be solved on each subsystem and a coupling QP to be solved by a coordinator is presented in [Bitlislioğlu et al. 2017]. Besides the multi-agent viewpoint, there also exist distributed interior point methods for large-scale parallel computing [Zavala et al. 2008]. For instance, expensive computations for forming the Schur complement of the Newton system can be parallelized among subsystems [Kang et al. 2014; Word et al. 2014]. Exemplary applications are vehicle coordination [Hult et al. 2022], building control [Bitlislioğlu et al. 2017], process systems [Zavala et al. 2008; Kang et al. 2014; Word et al. 2014], and Optimal Power Flow (OPF) [Engelmann et al. 2021b; Engelmann et al. 2022].

2.3.2 Distributed Gradient Projection

Distributed Gradient Projection (DGP) is one of the earliest algorithms to be proposed for nonlinear DMPC [Stewart et al. 2011]. According to the taxonomy adopted here, the method is essentially decentralized and proceeds as follows for solving condensed cooperative OCPs of the form

$$\min_{\boldsymbol{u}} V(\boldsymbol{u}) \quad \text{subject to} \quad \boldsymbol{u}_i \in \mathbb{U}_i \quad \text{for all} \quad i \in \mathcal{S}.$$

The nonlinear system dynamics are substituted into the twice-continuously differentiable non-convex centralized objective $V : \mathrm{R}^{n_u \cdot (N+1)} \to \mathbb{R}$ and the predicted input trajectories lie in the convex compact constraint sets $\mathbb{U}_i$. To accommodate this condensing, all subsystems have full knowledge of the centralized system dynamics. In each NMPC step, all subsystems are initialized with a feasible guess for the centralized input trajectory $\boldsymbol{u}$. Each subsystem performs an individual gradient projection and Armijo line search to compute a candidate step for $\boldsymbol{u}_i$. Then, a decrease condition for the centralized objective V is checked for the candidate updates. This can either be done on a central coordinator or, to obtain an essentially decentralized method, by communicating all candidate updates to all subsystems such that each subsystem checks the centralized decrease condition. The algorithm terminates if the decrease condition is met or else updates the step lengths for the individual subsystems.

Assumptions, feasibility, and stability DGP is applicable to cooperative OCPs with coupled nonlinear dynamics, coupled costs, and convex compact input constraints. A feasible initialization is required for the centralized input trajectory $\boldsymbol{u}$ and all subsequent optimizer iterates are feasible with respect to all constraints. The feasible guess can be set to the solution of the previous NMPC step. Thanks to the condensing of the state variables, this warm start is feasible even in the presence of model-plant mismatch or other disturbances. The method converges to a stationary point of the centralized objective and closed-loop stability follows from the feasible optimizer iterates for appropriate OCP designs, both with or without terminal state constraints [Stewart et al. 2011].

Implementation and applications The price to pay for these strong feasibility guarantees is that all subsystems must optimize over the centralized system dynamics, which can limit scalability and may violate privacy concerns. Moreover, the method either requires all-to-all communication or a coordinator for evaluating the decrease condition on V. DGP has been validated in mobile robot experiments with formation control [Rosenfelder et al. 2022] and cooperative object transportation [Ebel et al. 2024].

2.3.3 Augmented Lagrangian Alternating Direction Inexact Newton Method

ALADIN is a distributed method for partially separable NLPs and combines ideas from ADMM for decomposing expensive computations with elements of SQP methods for guaranteeing convergence in the presence of non-convex constraints [Houska et al. 2016]. In each ALADIN iteration, each subsystem first solves a small-scale non-convex NLP which is similar to the subsystem NLP to be solved in Step 3 of ADMM. Then, each subsystem approximates the Hessian of L_i and evaluates the Jacobian of the active constraints. These derivatives formulate a coupling QP which is solved on a central coordinator. If desired, the coordinator can further perform a globalization routine.

Assumptions, convergence, and stability If the augmented Lagrangian penalty parameter is sufficiently large, then ALADIN inherits the local convergence properties of inexact Newton methods and converges quadratically to regular KKT points [Houska et al. 2016]. If an additional globalization routine is employed, then ALADIN even converges in finite time under further mild technical assumptions [Houska et al. 2016, Theorem 2]. In the context of DMPC, these guarantees qualify ALADIN for a closed-loop stability analysis based on real-time iterations [Zanelli et al. 2021], a concept we will discuss in more detail in Chapter 4.

Variants, implementation, and applications While ALADIN was originally conceived for non-convex problems, the algorithm is also effective for linear-quadratic DMPC [Houska and Shi 2022] and there exist derivative-free variants for non-smooth problems [Houska and Jiang 2021].

The strong convergence guarantees come at the cost of centralized computation for solving the coupling QP. BL-ALADIN decomposes the coupling QP as a sparse linear system of equations to be solved using ADMM or d-CG, converting ALADIN into a decentralized or essentially decentralized algorithm, respectively [Engelmann et al. 2020]. Indeed, this decentralization via d-CG motivated the development of the bi-level algorithms d-ASM and d-IP. ALADIN and its variants have been applied to e.g. AC-OPF [Engelmann et al. 2019], smart grid control [Jiang et al. 2021], and solar parabolic trough plants [Chanfreut et al. 2023]. Open source implementations of the classical and BL-ALADIN variants can be found in the ALADIN-α toolbox [Engelmann et al. 2021a].

2.3.4 Further Algorithms

Further distributed bi-level algorithms for non-convex partially separable NLPs are the Bi-Level Sequential Convex Programming (BL-SCP) scheme in [Necoara et al. 2009], the Distributed SQP method in [Hult et al. 2016; Zanon et al. 2017; Hult et al. 2020], and the ELLADA algorithm [Tang and Daoutidis 2023]. Additional decentralized bi-level algorithms with convergence guarantees for non-convex constraints are the Optimality Tracking Splitting Algorithm (OTSA) for NLPs with polynomial objectives and constraint functions [Hours and Jones 2016] and the Two-Level Augmented Lagrangian Method (TL-ALM) with ADMM on the inner level from [Sun and Sun 2021; Sun and Sun 2023]. Notably, OTSA and TL-ALM exhibit local convergence guarantees for non-convex constraints but do *not* require a coordinator. OTSA assumes polynomial objectives and equality constraints and assigns subsystems to groups which execute expensive steps per iteration in sequence and not in parallel. TL-ALM covers more general problem settings, but requires each subsystem to solve an NLP in each inner ADMM iteration.

2.4 Algorithm Comparison

Tables 2.1 and 2.2 summarize key properties of the covered algorithms. We note that the convergence results and analyses presented in the given references often include technical assumptions which we omit here for brevity, especially in the case of non-convex optimization. Moreover, the algorithms are typically presented for problem formulations that differ slightly from NLP (2.1). For the sake of readability, Tables 2.1 and 2.2 present key results in the notation of this chapter, sometimes at the cost of mathematical inaccuracies compared to their original presentation. The tables thus provide an algorithmic overview with relevant properties in the context of DMPC, but we refer the reader to the individual references for detailed discussions.

As shown in Table 2.1, there exist numerous decentralized algorithms which enjoy convergence guarantees for linear-quadratic DMPC. Jacobi iterations produce feasible iterates, which simplifies the analysis of closed-loop properties, but complicates their implementation due to the requirement of feasible initializations. Thus, dual algorithms based on DD and ADMM allow for infea-

sible initializations at the cost of asymptotic consensus. From this point of view, d-ASM forms a compromise as it is feasible-side convergent with an essentially decentralized procedure for computing feasible initializations for OCPs without state constraints. Nonetheless, ADMM can be regarded as a prime candidate for solving the QPs arising in linear-quadratic DMPC, thanks to its amenability for infeasible warm-starts, linear convergence, decentralized implementation, and practical performance [Conte et al. 2012; Stomberg et al. 2022a].

The landscape of algorithms for non-convex NLPs is more nuanced, see Table 2.2. This is caused by the inherent difficulty associated with non-convexity, especially by non-convex constraints. As a result, many algorithms rely on a centralized coordinator, e.g. ALADIN, BL-SCP, Distributed SQP, and ELLADA. Important algorithmic features which typically necessitate either centralized or essentially decentralized communication are dynamic step sizes, e.g. line search procedures, because all subsystems must usually agree on the same step size in a given iteration or because a centralized merit function is employed. Decentralized algorithms with convergence guarantees for nonlinear DMPC date back at least to DGP. From a feasibility point of view, this method can be seen as the NLP counterpart to Jacobi iterations: all iterates are feasible, but a required feasible initial guess limits the applicability. This drawback is overcome by the decentralized bi-level algorithms OTSA, BL-ALADIN/d-ADMM, and TL-ALM. The latter two were first applied to AC-OPF problems [Engelmann et al. 2020; Sun and Sun 2021], where the computing hardware can be assumed to be sufficiently strong such that the solution of NLPs on each subsystem is not a major concern. However, in the context of nonlinear DMPC, solving NLPs on embedded hardware can form a computational bottleneck which limits real-time feasibility [Van Parys and Pipeleers 2017]. OTSA only requires projected gradient steps to be computed on a subsystem level, which is computationally less expensive than solving NLPs. On the other hand, BL-ALADIN and TL-ALM enjoy convergence guarantees for more general problems as OTSA assumes that the system dynamics are polynomial.

Compared to the widespread attention received by ADMM for linear-quadratic DMPC, only few results investigate the performance of the above algorithms for nonlinear DMPC in simulations or real-time experiments. Thus, we conclude this comparison by observing that a decentralized algorithm for nonlinear DMPC with convergence guarantees, small subsystem computation footprint, *and* experimental validation appears to be unavailable.

2.5 Adversarial Example: What Can Go Wrong With Infeasible Iterates?

A key feature discussed above is an algorithm's ability to generate feasible iterates. While feasible-side convergent algorithms improve closed-loop stability, their decentralized initialization is impractical. Therefore, the rest of this thesis discusses real-time algorithms with infeasible iterates. This section recalls an adversarial example to illustrate the dangers associated with infeasibil-

Table 2.1: Comparison of optimization algorithms for linear-quadratic DMPC. Table adapted from [Stomberg et al. 2022a].

Method	Communication	Assumptions on QP	Proven convergence	Feasible iterates	Further reading
ADMM	decentralized	convex	linear in $w = (z, \gamma/\rho)$	$y_i^k \in \mathbb{Z}_i$, $z^k \in \mathbb{E}$	[Boyd et al. 2011; Bestler and Graichen 2019]
d-ASM	ess. decentralized	strictly convex, LICQ for all z^k	finite-time	$z^k \in \mathbb{Z} \cap \mathbb{E}$	[Stomberg et al. 2021; Stomberg et al. 2022a]
DD	decentralized	f^{QP} strongly convex, $\mathbb{Z}^{\text{QP}}$ compact	$O(1/k)$ in $q(\lambda^\star) - q(\lambda^k)$	$z_i^k \in \mathbb{Z}_i$	[Braun and Grüne 2018; Köhler et al. 2019]
DD-FGM	decentralized	f^{QP} strongly convex, $\mathbb{Z}^{\text{QP}}$ compact	$O(1/k^2)$ in $q(\lambda^\star) - q(\lambda^k)$	$z_i^k \in \mathbb{Z}_i$	[Conte et al. 2012; Giselsson et al. 2013]
Jacobi	decentralized	convex	$\lim_{k\to\infty} f(z^k) = c$	$z^k \in \mathbb{Z} \cap \mathbb{E}$	[Doan et al. 2017; Groß and Stursberg 2013]

Table 2.2: Comparison of optimization algorithms for nonlinear DMPC.

Method	Communication	Assumptions on NLP	Proven convergence	Feasible iterates	Further reading
ALADIN	distributed	regular $p^\star$	finite-time	$y^k \in \mathbb{Z}$	[Houska et al. 2016; Jiang et al. 2021]
BL-ALADIN/ ADMM	decentralized	regular $p^\star$ & iterates	quadratic	$y^k \in \mathbb{Z}$	[Engelmann et al. 2020; Engelmann et al. 2021a]
BL-ALADIN/ d-CG	ess. decentralized	regular $p^\star$ & iterates	quadratic	$y^k \in \mathbb{Z}$	[Engelmann 2020]
BL-SCP	distributed	convex f & h, LICQ, str. comp.	$z \to z^\star$ linearly	no	[Necoara et al. 2009]
DGP	ess. decentralized	$\mathbb{U}$ compact & convex, no $\mathbb{X}$	$\nabla V(\boldsymbol{u}^k) \to 0$	$z^k \in \mathbb{Z} \cap \mathbb{E}$	[Stewart et al. 2011; Rosenfelder et al. 2022]
d-IP	ess. decentralized	regular KKT point $p^\star$	$p^k \to p^\star$ superlinearly	no	[Engelmann et al. 2021b; Engelmann et al. 2022]
Distributed SQP	distributed	Lipschitz KKT Jacobian	$p^k \to p^\star$ globally	no	[Hult et al. 2016; Zanon et al. 2017]
ELLADA	distributed	$L_\rho^k \leq \bar{L}$ for all k	$p^k \to p^\star$	block-wise	[Tang and Daoutidis 2023]
OTSA	decentralized	polynomial f & g, regular $p^\star$	$z \to z^\star$ sublinearly	no	[Hours and Jones 2016]
TL-ALM	decentralized	compact $\mathbb{Z}$, convex & compact $\mathbb{E}$	yes, global & local	block-wise	[Sun and Sun 2021; Sun and Sun 2023]

ity [Stomberg et al. 2022a]. Consider the three coupled first-order systems

$$\dot{x}_1(t) = u_1(t), \quad \dot{x}_2(t) = x_1(t) + 4x_2(t), \quad \dot{x}_3(t) = x_2(t) + 4x_3(t), \quad x(0) = (1, 2, 3)$$

with constraints $-1000 \leq u_1 \leq 1000$ and $-200 \leq x_i \leq 200$ for all $i \in \mathcal{S}$. While the centralized system is controllable, subsystems two and three are unstable and must be stabilized through u_1. We discretize the systems with an exact zero-order hold at a sampling interval $\delta = 0.040$ and design the cooperative OCP (1.3) with horizon $N = 50$ and decoupled stage costs $\ell_1(x_1, u_1) \doteq 5x_1^2 + 0.5u_1^2$, $\ell_2(x_2) \doteq 5x_2^2$, and $\ell_3(x_3) \doteq 5x_3^2$. Moreover, we guarantee stability for feasible iterates by imposing the terminal constraint $x[N] = 0$ in the OCP.

We compare d-ASM, DD, DD-FGM, ADMM, d-IP, and Jacobi Iterations with the following settings. For d-ASM, we set $\varepsilon = 10^{-6}$, specify $r < 10^{-8}$ when terminating d-CG, and compute a feasible initial guess as described above. We select the ADMM penalty parameter as $\rho = 60$, choose the DD step length $c = 0.99 \cdot 2\mu_c/\|E\|_2^2$, and set the Lipschitz constant for DD-FGM to $L = \|EH^{-0.5}\|_2^2$ [Richter et al. 2011]. For d-IP, we set $(c_1, \gamma, \beta, \eta, \varepsilon) = (1, 1.01, 2, 1.01, 10^{-7})$ and tune to get $\theta = 0.1$ and $\delta^0 = 1$. We obtain a feasible initialization for Jacobi Iterations by solving a feasibility problem with the centralized OCP constraints and zero objective. Then, we set $\omega_i = 1/|\mathcal{S}|$ for all $i \in \mathcal{S}$.

Figure 2.2 shows the convergence of the six methods to the unique OCP minimizer $z^\star$ after a cold start with $z^0 = 0$ in the first MPC step. Here and in Figures 2.3–2.4, the iteration count for the bi-level algorithms d-ASM and d-IP refers to the inner (d-CG) iterations. Jacobi Iterations, d-ASM, and d-IP are the fastest to converge to high accuracy. However, note that Jacobi Iterations started from a solution found with a centralized QP solver. DD, DD-FGM, and ADMM struggle to achieve high accuracy in z but converge to the optimal control input in a few hundred iterations. Figure 2.3 illustrates the closed-loop control performance and shows the final system state $x(t_f)$ after 125 MPC steps versus the number of optimizer itertions per control step. Because we warm start the algorithms with the solutions obtained in the prior MPC step shifted by the time index, ADMM and DD-FGM stabilize the system with less than 500 iterations per control step. However, Jacobi Iterations, d-ASM, and d-IP require far less iterations, showcasing the benefits of feasible iterates or, in the case of d-IP, superlinear convergence. Indeed, Figure 2.4 highlights the dangers of infeasibility as 70 ADMM iterations per MPC step are stabilizing, but 69 iterations are not.

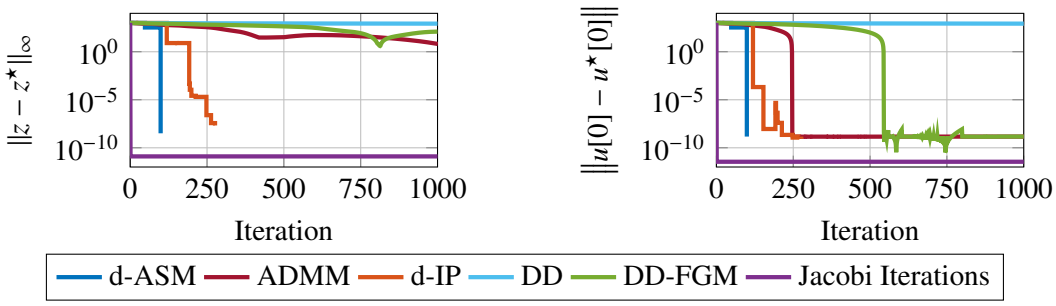


Figure 2.2: Convergence to the OCP minimizer in the first MPC step. Figure adapted from [Stomberg et al. 2022a] which has been published under the CC BY 4.0 license, cf. `https://creativecommons.org/licenses/by/4.0/`.

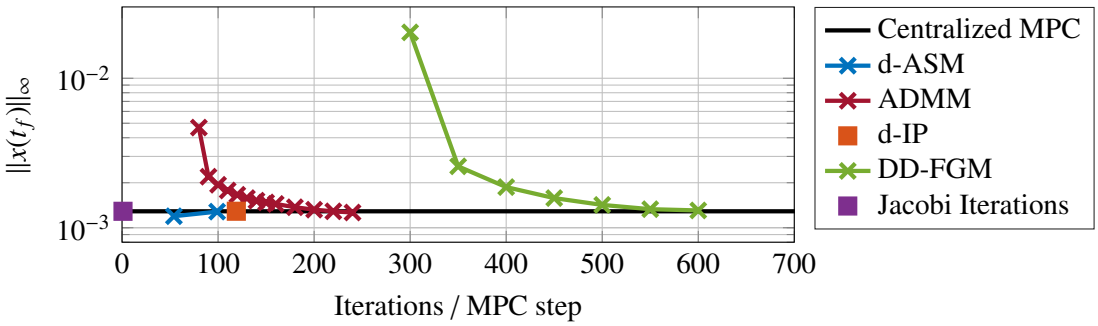


Figure 2.3: Final state $x(t_f)$ after 125 MPC steps depending on the number of optimizer iterations per control step. Figure adapted from [Stomberg et al. 2022a] which has been published under the CC BY 4.0 license.

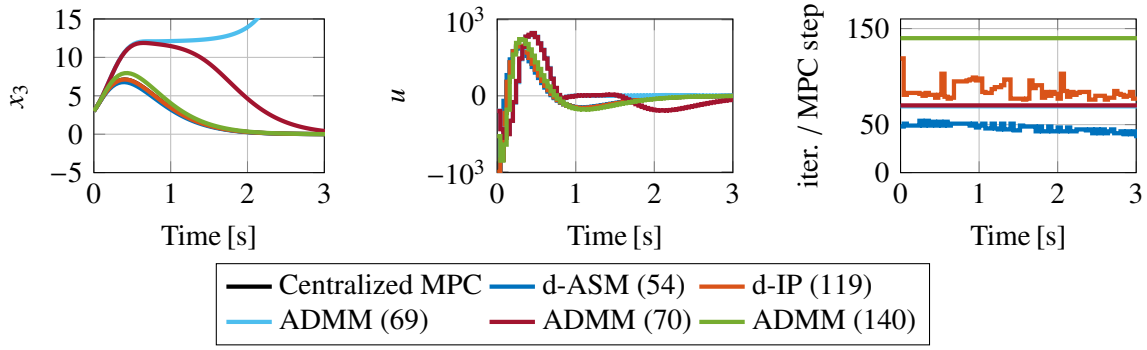


Figure 2.4: Closed-loop trajectories for different algorithms. The maximum number of optimizer iterations per control step is given in parentheses. Figure adapted from [Stomberg et al. 2022a] which has been published under the CC BY 4.0 license.

2.6 Summary

This chapter has summarized the state of the art on optimization algorithms for cooperative DMPC. We have presented the main ideas underlying dual decomposition, ADMM, Jacobi iterations, d-ASM, and d-IP and we have discussed their advantages and disadvantages for DMPC. Moreover, we have summarized key assumptions and theoretical properties of DGP, ALADIN, and further bi-level algorithms based on augmented Lagrangian or second-order methods in the context of nonlinear DMPC. A numerical example with unstable and uncontrollable subsystems demonstrates the risks of early termination and the associated lack of consensus. The example illustrates the importance of system partitioning in CPSoS as controllability of the centralized system suffices in theory to obtain stability, but not necessarily in practical implementations subject to real-time constraints. Despite its asymptotic convergence towards consensus, ADMM appears to be among the most promising methods for linear-quadratic DMPC thanks to its theoretical guarantees *and* observed performance. In contrast, it is less well understood which algorithms perform well for nonlinear DMPC. Especially decentralized algorithms with guarantees in the presence of non-convex constraints are scarce, many of which require to solve NLPs on the subsystem level.

Consequently, there is a need for decentralized algorithms with, (a), guaranteed convergence in open and closed loop, (b), low computational cost to facilitate embedded implementations; and, (c), experimental validation. To address this gap, the next chapter presents an efficient algorithm based on ADMM and proves convergence, even for non-convex constraints.

3 Decentralized Sequential Quadratic Programming

This chapter presents two bi-level algorithms based on SQP and ADMM for solving non-convex partially separable NLPs. We first present the core ideas of the approach and provide a basic algorithmic variant using a tailored stopping criterion for ADMM. We then prove local convergence by combining the Newton-type analysis of SQP schemes with the ADMM convergence for convex QPs. The active set requires special attention in the convergence analysis and we illustrate possible pitfalls for a small-scale example in Section 3.3. Moreover, we present a second algorithmic variant which allows for a simpler inequality constraint analysis. Finally, we analyze the performance of both proposed algorithms for an optimal power flow problem.

The basic algorithmic framework, the main steps in the convergence analysis, and parts of the numerical results have appeared in the conference paper [Stomberg et al. 2022b]. The example illustrating the active set stability of ADMM has appeared, in shortened form, in the revised version of [Stomberg et al. 2022b] that is avilable online.[1] The two-block algorithm in Section 3.4 has not been published before.

3.1 Bi-level Decentralized SQP and ADMM

We recall the partially separable NLP (2.1)

$$\min_{z} \sum_{i\in\mathcal{S}} f_i(z_i) \tag{3.1a}$$

$$\text{subject to} \quad g_i(z_i) = 0 \mid \nu_i \quad \forall i \in \mathcal{S}, \tag{3.1b}$$

$$h_i(z_i) \leq 0 \mid \mu_i \quad \forall i \in \mathcal{S}, \tag{3.1c}$$

$$\sum_{i\in\mathcal{S}} E_i z_i = b \mid \lambda. \tag{3.1d}$$

Assumption 3.1 (Differentiability of the NLP functions)**.** *The functions f_i, g_i, and h_i are three times continuously differentiable for all $i \in \mathcal{S}$.*[2] □

Centralized SQP schemes are well-established methods which repeateadly generate and solve convex QP approximations of NLP (3.1) [Boggs and Tolle 1995]. The fast local convergence behavior, cf. Appendix A.4, and the amenability for warm starts have encouraged their use RTI schemes [Diehl et al. 2002a; Gros et al. 2020]. This motivates the use of SQP methods in DMPC

[1] `https://arxiv.org/abs/2204.08786`

[2] The algorithms presented in this chapter can also be applied if the functions f_i, g_i, and h_i are only twice continuously differentiable. However, we here assume three times continuous differentiability to invoke the Basic Sensitivity Theorem A.3 from Appendix A.2 when proving local convergence.

and we present a decentralized SQP scheme which exploits the structure of the partially separable NLP (3.1) by solving the arising subproblem QPs in decentralized fashion via ADMM. The method owes its name to the bi-level structure with SQP iterations on the outer level and ADMM iterations on the inner level. We index outer iterations by k and inner iterations by l. Thus, we denote SQP iterates by a superscript $\cdot^k$ and the ADMM iterates in the $k-$th SQP step by a superscript $\cdot^{k,l}$.

Recall the Lagrangian of NLP (3.1),

$$L(z,\nu,\mu,\lambda) \doteq \sum_{i\in\mathcal{S}} L_i(z_i,\nu_i,\mu_i,\lambda) - \lambda^\top b \quad \text{with} \quad L_i(z_i,\nu_i,\mu_i,\lambda) \doteq f_i(z_i) + \nu_i^\top g_i(z_i) + \mu_i^\top h_i(z_i) + \lambda^\top E_i z_i.$$

Consider a convex approximation of NLP (3.1) at a point $p^k \doteq (z^k,\nu^k,\mu^k,\lambda^k) \in \mathbb{R}^{n_p}$,

$$\min_z \sum_{i\in\mathcal{S}} f_i^{\mathrm{QP},k}(z_i) \tag{3.2a}$$

$$\text{subject to} \quad g_i^k + \nabla g_i^{k\top}(z_i - z_i^k) = 0 \mid \nu_i \quad \forall i \in \mathcal{S}, \tag{3.2b}$$

$$h_i^k + \nabla h_i^{k\top}(z_i - z_i^k) \le 0 \mid \mu_i \quad \forall i \in \mathcal{S}, \tag{3.2c}$$

$$\sum_{i\in\mathcal{S}} E_i z_i = b \mid \lambda. \tag{3.2d}$$

The objective functions are defined as $f_i^{\mathrm{QP},k} \doteq (z_i - z_i^k)^\top H_i^k (z_i - z_i^k)/2 + \nabla f_i^{k\top}(z_i - z_i^k)$ where $H_i^k \approx \nabla^2_{z_i z_i} L_i(z_i^k,\nu_i^k,\mu_i^k)$ are Hessian approximations for all $i \in \mathcal{S}$. The symbols g_i^k and ∇g_i^k in QP (3.2) are shorthands for $g_i(z_i^k)$ and $\nabla g_i(z_i^k)$, respectively and the same holds for functions f_i and h_i. As we will prove in this section, QP (3.2) has a unique centralized KKT point if p^k is close to a regular KKT point $p^\star$. We denote the KKT point of QP (3.2) as $p^{k,\star} = (z^{k,\star},\nu^{k,\star},\mu^{k,\star},\lambda^{k,\star})$, where $z^{k,\star} = (z_1^{k,\star},\dots,z_S^{k,\star})$, $\nu^{k,\star} = (\nu_1^{k,\star},\dots,\nu_S^{k,\star})$, and $\mu^{k,\star} = (\mu_1^{k,\star},\dots,\mu_S^{k,\star})$.

To apply ADMM, we introduce a local decision variable $y_i \in \mathbb{R}^{n_i}$ for each $i \in \mathcal{S}$ and we rewrite QP (3.2) in two-block form as

$$\min_{y,z} \sum_{i\in\mathcal{S}} f_i^{\mathrm{QP},k}(y_i) \tag{3.3a}$$

$$\text{subject to} \quad g_i^k + \nabla g_i^{k\top}(y_i - z_i^k) = 0 \mid \nu_i \quad \forall i \in \mathcal{S}, \tag{3.3b}$$

$$h_i^k + \nabla h_i^{k\top}(y_i - z_i^k) \le 0 \mid \mu_i \quad \forall i \in \mathcal{S}, \tag{3.3c}$$

$$\sum_{i\in\mathcal{S}} E_i z_i = b \mid \lambda, \tag{3.3d}$$

$$y_i - z_i = 0 \mid \gamma_i \quad \forall i \in \mathcal{S}. \tag{3.3e}$$

We define the centralized variables and components $y \doteq (y_1,\dots,y_S)$, $H^k \doteq \mathrm{diag}(H_1^k,\dots,H_S^k)$, $\nabla f^k \doteq (\nabla f_1^k,\dots,\nabla f_S^k)$, and $\mathbb{Z}^k \doteq \mathbb{Z}_1^k \times \cdots \times \mathbb{Z}_S^k$ with

$$\mathbb{Z}_i^k \doteq \left\{ y_i \in \mathbb{R}^{n_i} \;\middle|\; \begin{matrix} g_i^k + \nabla g_i^{k\top}(y_i - z_i^k) = 0 \\ h_i^k + \nabla h_i^{k\top}(y_i - z_i^k) \le 0 \end{matrix} \right\}.$$

Moreover, we define the augmented Lagrangian of QP (3.3) as

$$L_\rho^k(y,z,\gamma) \doteq \sum_{i\in\mathcal{S}} L_{\rho,i}^k(y_i,z_i,\gamma_i) \doteq \sum_{i\in\mathcal{S}} \left(f_i^{\mathrm{QP},k}(y_i) + \gamma_i^\top (y_i - z_i) + \frac{\rho}{2}\|y_i - z_i\|_2^2 \right).$$

The ADMM iterations for solving QP (3.3) read

$$(y_i^{k,l+1}, \nu_i^{k,l+1}, \mu_i^{k,l+1}) \leftarrow \min_{y_i\in\mathbb{Z}_i^k} L_{\rho,i}^k(y_i, z_i^{k,l}, \gamma_i^{k,l}) \qquad \forall i \in \mathcal{S} \tag{3.4a}$$

$$z^{k,l+1} = \operatorname*{argmin}_{z\in\mathbb{E}} L_\rho^k(y^{k,l+1}, z, \gamma^{k,l}) \tag{3.4b}$$

$$\gamma_i^{k,l+1} = \gamma_i^{k,l} + \rho(y_i^{k,l+1} - z_i^{k,l+1}) \qquad \forall i \in \mathcal{S}. \tag{3.4c}$$

The above notation indicates that Step (3.4a) updates the decision variables y_i as well as the subsystem constraint multipliers ν_i and μ_i.

Let $\|\cdot\|$ refer to any norm on $\mathbb{R}^n$ or its induced matrix norm, respectively. Algorithm 3.1 summarizes the bi-level dSQP method. The scheme starts from a primal-dual initialization $p^0 = (z^0, \nu^0, \mu^0, \lambda^0)$ in Step 1, constructs QP (3.2) in Step 4, applies ADMM in Steps 7–9, and then updates the SQP iterates in Step 12. To ensure the ADMM solution in the k–th SQP step is sufficiently accurate, we enforce a tailored stopping criterion in Step 6. This inner stopping criterion and the outer termination criterion in Step 2 are based on the maps

$$\tilde{F}(p) \doteq \begin{bmatrix} \nabla_{z_1} L_1(z_1,\nu_1,\mu_1,\lambda) \\ g_1(z_1) \\ \vdots \\ \nabla_{z_S} L_S(z_S,\nu_S,\mu_S,\lambda) \\ g_S(z_S) \\ \sum_{i\in\mathcal{S}} E_i z_i - b \end{bmatrix} \quad \text{and} \quad F(p) \doteq \begin{bmatrix} \nabla_{z_1} L_1(z_1,\nu_1,\mu_1,\lambda) \\ g_1(z_1) \\ \min(-h_1(z_1),\mu_1) \\ \vdots \\ \nabla_{z_S} L_S(z_S,\nu_S,\mu_S,\lambda) \\ g_S(z_S) \\ \min(-h_S(z_S),\mu_S) \\ \sum_{i\in\mathcal{S}} E_i z_i - b \end{bmatrix}. \tag{3.5}$$

Here,

$$\min(x,y) \doteq (\min([x]_1,[y]_1), \ldots, \min([x]_n,[y]_n))$$

denotes the vector of componentwise minima for two vectors $x, y \in \mathbb{R}^n$. Moreover, F^k, $\tilde{F}^k$, and $\nabla\tilde{F}^k$ in Algorithm 3.1 are abbreviations for $F(p^k)$, $\tilde{F}(p^k)$, and $\nabla\tilde{F}(p^k)$. Compared to $\tilde{F}$, the map F additionally includes the complementarity conditions $\min(-h_i(x_i),\mu_i)$ for all $i \in \mathcal{S}$.

Remark 3.1 (Decentralized implementation)**.** *An important advantage of dSQP is its amenability to decentralized implementation. To decentralize Algorithm 3.1, one must decompose the evaluation of the stopping criterion in Step 6 and the z update of ADMM in Step 8. The former can be achieved by choosing $\|\cdot\|_\infty$ in the stopping criterion, because $z^{k,l} \in \mathbb{E}$ for all $k, l \geq 0$. To decentralize ADMM, we implement the z update as an averaging step, if NLP* (3.1) *is a consensus problem as specified in Definition 2.2, cf. Remark 2.1.* □

Algorithm 3.1 Bi-level decentralized SQP for solving (3.1) [Stomberg et al. 2022b]

1: SQP initialization: $k = 0$, $(z_i^0, \nu_i^0, \mu_i^0, \gamma_i^0 = E_i^\top \lambda^0)$ for all $i \in \mathcal{S}$, $\eta^0 < 1$, ϵ
2: **while** $\|F^k\| \not\leq \epsilon$ **do**
3: compute ∇f_i^k, g_i^k, ∇g_i^k, h_i^k, ∇h_i^k, and H_i^k for all $i \in \mathcal{S}$
4: construct $\tilde{F}^k$ and $\nabla \tilde{F}^k$
5: ADMM initialization: $l = 0$, $(z_i^{k,0}, \gamma_i^{k,0}, \nu_i^{k,0}, \mu_i^{k,0}) = (z_i^k, \gamma_i^k, \nu_i^k, \mu_i^k)$ for all $i \in \mathcal{S}$
6: **while** $l = 0$ **or** $\|\tilde{F}^k + \nabla \tilde{F}^{k\top}(p^{k,l} - p^k)\| \not\leq \eta^k \|\tilde{F}^k\|$ **do**
7: $\left(y_i^{k,l+1}, \nu_i^{k,l+1}, \mu_i^{k,l+1}\right) \leftarrow \min_{y_i \in \mathbb{Z}_i^k} L_{\rho,i}^k(y_i, z_i^{k,l}, \gamma_i^{k,l})$ for all $i \in \mathcal{S}$
8: $z^{k,l+1} = \operatorname{argmin}_{z \in \mathbb{B}} L_\rho^k(y^{k,l+1}, z, \gamma^{k,l})$
9: $\gamma_i^{k,l+1} = \gamma_i^{k,l} + \rho(y_i^{k,l+1} - z_i^{k,l+1})$ for all $i \in \mathcal{S}$
10: $l \leftarrow l + 1$
11: **end while**
12: $(z_i^{k+1}, \nu_i^{k+1}, \mu_i^{k+1}, \gamma_i^{k+1}) = (z_i^{k,l}, \nu_i^{k,l}, \mu_i^{k,l}, \gamma_i^{k,l})$ for all $i \in \mathcal{S}$
13: choose $\eta^{k+1} \leq \eta^k$
14: $k \leftarrow k + 1$
15: **end while**
16: **return** z_i^k for all $i \in \mathcal{S}$

Remark 3.2 (Dual iterates of the coupling constraints)**.** *If Step 8 is implemented via the ADMM averaging procedure as discussed in Remark 2.1, then Algorithm 3.1 does not compute the coupling constraint Lagrange multipliers λ^k. However, only $E_i^\top \lambda$ is required to evaluate Steps 2, 4, and 6 such that we can exploit the fact that $E_i^\top \lambda^k = \gamma_i^k$. For more details, see Lemma 2.1.* □

3.2 Convergence Analysis

The centralized SQP convergence analysis in Theorem A.6 exploits the local equivalence to Newton's method and serves as a blueprint for the analysis presented in this section. In order to improve the numerical performance of dSQP, we terminate ADMM early on the inner level. The resulting approximate nature of the SQP steps requires additional attentation in the analysis and the convergence proof therefore follows in two stages. First, we derive convergence on the outer level based on the stopping criterion in Step 2 of Algorithm 3.1. Then, we rely on the ADMM convergence Theorem A.8 from Appendix A.5 on the inner level. Figure 3.1 summarizes key steps of the proof and visualizes intermediate convergence radii to assist the reader during the remainder of this section.

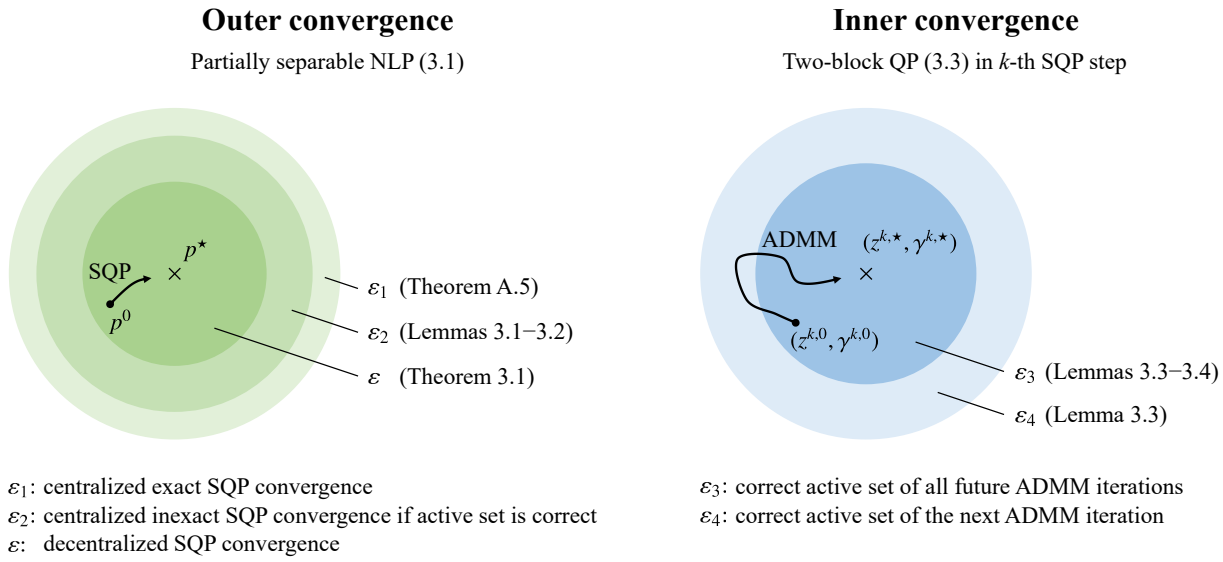


Figure 3.1: Summary of intermediate results for proving local convergence of dSQP. As we show below, ADMM converges r-linearly in $\|(z^{k,l}, y^{k,l}) - (z^{k,\star}, \gamma^{k,\star})\|$ and the inexact Newton-type stopping criterion to be derived for the SQP steps ensures that ADMM terminates inside $\mathcal{B}((z^{k,\star}, \gamma^{k,\star}), \varepsilon_3)$ to stay at the correct active set.

3.2.1 Outer Convergence

Let $z^\star$ be a local minimum of NLP (3.1) and denote the active set of subsystem i at $z_i^\star$ as $\mathcal{A}_i(z_i^\star) \doteq \{j \in \{1, \dots, n_{h_i}\} \mid [h_i(z_i^\star)]_j = 0\}$.

Assumption 3.2 (Regular KKT point)**.** *The point $p^\star = (z^\star, \nu^\star, \mu^\star, \lambda^\star)$ is a KKT point of NLP* (3.1) *which, for all $i \in \mathcal{S}$, satisfies*

i) $h_i(z_i^\star) + \mu_i^\star \neq 0$ *(strict complementarity),*

ii) $z_i^\top \nabla^2_{z_i z_i} L_i(z_i^\star, \nu_i^\star, \mu_i^\star) z_i > 0$ *for all* $z_i \neq 0$ *with* $\nabla g_i(z_i^\star)^\top z_i = 0$ *(stronger SOSC)*[3]

Furthermore, the matrix

$$\begin{bmatrix} \nabla g_1(z_1^\star)^\top & & \\ & \ddots & \\ & & \nabla g_S(z_S^\star)^\top \\ \left[\nabla h_1(z_1^\star)^\top\right]_{\mathcal{A}_1(z_1^\star)} & & \\ & \ddots & \\ & & \left[\nabla h_S(z_S^\star)^\top\right]_{\mathcal{A}_S(z_S^\star)} \\ E_1 & \dots & E_S \end{bmatrix}$$

[3]This condition is slightly stronger than the Second-Order Sufficient Conditions (SOSC), because we exclude the conditions $\nabla[h_i(z_i)]_j^\top z_i = 0$ for all $j \in \mathcal{A}_i(z_i^\star)$ and $E_i z_i = 0$, cf. (A.4).

has full row rank, i.e., $z^\star$ satisfies the Linear Independence Constraint Qualification (LICQ).

□

We denote the open ε neighborhood around the KKT point by $\mathcal{B}(p^\star, \varepsilon) \doteq \{p \in \mathbb{R}^{n_p} \,|\, \|p - p^\star\| < \varepsilon\}$. The centralized SQP convergence Theorem A.6 is based on the fact that the sequence of exact SQP iterates $\{p^k\}$ satisfies the Newton iteration $F^k + \nabla F^k(p^{k+1} - p^k) = 0$ if $p^k \in \mathcal{B}(p^\star, \varepsilon_1)$ for some $\varepsilon_1 > 0$ [Geiger and Kanzow 2002]. We relax this requirement and, inspired by [Curtis et al. 2014], instead impose the inexact Newton stopping criterion [Dembo et al. 1982]

$$\left\|F^k + \nabla F^{k\top}(p^{k+1} - p^k)\right\| \leq \eta^k \left\|F^k\right\| \tag{3.6}$$

with $0 < \eta^k < 1$.

Lemma 3.1 (Local convergence with inexact SQP steps)**.** *Suppose Assumption 3.1 holds and let $p^\star$ denote a KKT point of NLP* (3.1) *which satisfies Assumption 3.2. Form QP* (3.2) *with the Hessian $H_i^k = \nabla^2_{z_i z_i} L_i(z_i^k, v_i^k, \mu_i^k)$ for all $i \in \mathcal{S}$. Let the sequence $\{p^k\}$ generated by Algorithm 3.1 satisfy the inexact Newton stopping criterion* (3.6) *for all $k \geq 0$. Then there exist constants $\varepsilon_2 > 0$ and $\eta \in (0, 1)$ such that the following holds for all $p^0 \in \mathcal{B}(p^\star, \varepsilon_2)$:*

i) If $\eta^k \leq \eta$ for all $k \geq 0$, then the sequence $\{p^k\}$ converges q-linearly to $p^\star$.

ii) If additionally $\eta^k \to 0$, then the convergence rate is q-superlinear.

iii) If additionally $\eta^k = O(\|F^k\|)$, then the convergence rate is q-quadratic. □

Proof. Due to the centralized SQP convergence Theorem A.6 from Appendix A.4, we can choose $0 < \varepsilon_2 \leq \varepsilon_1$ such that $\nabla F(p)$ is regular inside $\mathcal{B}(p^\star, \varepsilon_2)$. To obtain statement i), we can therefore adapt [Dembo et al. 1982, Theorem 2.3] which proves the following: For any $a \in (\eta, 1)$, there exists $\varepsilon_2 > 0$ such that, if $p^0 \in \mathcal{B}(p^\star, \varepsilon_2)$, then the sequence $\{p^k\}$ converges linearly to $p^\star$ in the sense that

$$\left\|\nabla F(p^\star)^\top (p^{k+1} - p^\star)\right\| \leq a \left\|\nabla F(p^\star)^\top (p^k - p^\star)\right\| \tag{3.7}$$

for all $k \geq 0$. Similar to [Morini 1999, Equation 7], we obtain, for all $k \geq 0$,

$$\begin{aligned}
\left\|p^{k+1} - p^\star\right\| &= \left\|\left(\nabla F(p^\star)^\top\right)^{-1} \nabla F(p^\star)^\top (p^{k+1} - p^\star)\right\| \\
&\leq \left\|\left(\nabla F(p^\star)^\top\right)^{-1}\right\| \cdot \left\|\nabla F(p^\star)^\top (p^{k+1} - p^\star)\right\| \\
&\leq \left\|\left(\nabla F(p^\star)^\top\right)^{-1}\right\| \cdot a \left\|\nabla F(p^\star)^\top (p^k - p^\star)\right\| \\
&\leq \left\|\left(\nabla F(p^\star)^\top\right)^{-1}\right\| \cdot \left\|\nabla F(p^\star)^\top\right\| \cdot a \|p^k - p^\star\| \\
&= \operatorname{cond}(\nabla F(p^\star)^\top) \cdot a \left\|p^k - p^\star\right\|.
\end{aligned}$$

We have used inequality (3.7) to arrive at the third row and defined

$$\operatorname{cond}(\nabla F(p^\star)^\top) \doteq \left\|\left(\nabla F(p^\star)^\top\right)^{-1}\right\| \cdot \left\|\nabla F(p^\star)^\top\right\|$$

as the condition number of $\nabla F(p^\star)^\top$. Choosing η, a, and ε_2 sufficiently small yields q-linear convergence with the contraction factor $c = \text{cond}(\nabla F(p^\star)^\top) \cdot a < 1$. Since $\{p^k\}$ converges to $p^\star$, statements ii) and iii) follow from [Dembo et al. 1982, Corollary 3.5]. ■

Observe that the proof of Lemma 3.1 shows that η must be chosen small if $\text{cond}(F(p^\star)^\top)$ is large to guarantee linear convergence, i.e., ill-conditioned problems require more accurate ADMM solutions.

The inexact Newton stopping criterion (3.6) guarantees local convergence and is well-defined if $p^0 \in \mathcal{B}(p^\star, \varepsilon_2)$. However, in practice it is often unknown whether $p^0 \in \mathcal{B}(p^\star, \varepsilon_2)$. The block rows $\min\left(-h_i(z_i^k), \mu_i^k\right)$ of F are not necessarily differentiable outside $\mathcal{B}(p^\star, \varepsilon_2)$ and thus it may not be possible to evaluate (3.6) outside $\mathcal{B}(p^\star, \varepsilon_2)$. To address this differentiability issue, we propose a modified stopping criterion which is equivalent to (3.6) if ADMM returns an inexact solution with the correct active set, but which can be evaluated for all $p \in \mathbb{R}^{n_p}$. The modified criterion reads

$$\left\|\tilde{F}^k + \nabla\tilde{F}^{k\top}(p^{k+1} - p^k)\right\| \leq \eta^k \left\|\tilde{F}^k\right\|, \tag{3.8}$$

where $\tilde{F}(p)$ is defined in (3.5) and

$$\nabla\tilde{F}^{k\top} = \begin{bmatrix} \nabla^2_{z_1 z_1} L_1^k & \dots & 0 & \nabla g_1^k & \dots & 0 & \nabla h_1^k & \dots & 0 & E_1^\top \\ \nabla g_1^{k\top} & \dots & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & \nabla^2_{z_S z_S} L_S^k & 0 & \dots & \nabla g_S^k & 0 & \dots & \nabla h_S^k & E_S^\top \\ 0 & \dots & \nabla g_S^{k\top} & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ E_1 & \dots & E_S & 0 & \dots & 0 & 0 & \dots & 0 & 0 \end{bmatrix}.$$

Observe that $\tilde{F}$ does not include the block rows $\min(-h_i(z_i), \mu_i)$ to avoid potential differentiability issues outside $\mathcal{B}(p^\star, \varepsilon_2)$. Omitting these block rows is not critical if ADMM produces inexact QP solutions with the correct active set, as we show next.

Lemma 3.2 (Modified stopping criterion)**.** *Suppose Assumption 3.1 holds, let $p^\star$ denote a KKT point which satisfies Assumption 3.2 and let $p^k \in \mathcal{B}(p^\star, \varepsilon_1)$ with $\varepsilon_1 > 0$ from Theorem A.6. Form QP* (3.2) *with the Hessian $H_i^k = \nabla^2_{z_i z_i} L_i(z_i^k, \nu_i^k, \mu_i^k)$ for all $i \in \mathcal{S}$.*

If $p^{k+1} = (z^{k+1}, \nu^{k+1}, \mu^{k+1}, \lambda^{k+1})$ lies at the same active set as $p^\star$, i.e., if

$$[h_i^k]_j + [\nabla h_i^{k\top}(z_i^{k+1} - z_i^k)]_j = 0, \ \forall j \in \mathcal{A}_i(z_i^\star), \quad \forall i \in \mathcal{S}, \tag{3.9a}$$

$$[\mu_i^{k+1}]_j = 0, \ \forall j \in \mathcal{I}_i(z_i^\star), \quad \forall i \in \mathcal{S}, \tag{3.9b}$$

then

$$\left\|\tilde{F}^k + \nabla\tilde{F}^{k\top}(p^{k+1} - p^k)\right\| \leq \eta^k \left\|\tilde{F}^k\right\| \implies \left\|F^k + \nabla F^{k\top}(p^{k+1} - p^k)\right\| \leq \eta^k \left\|F^k\right\|.$$

Proof. Strict complementarity at $p^\star$ implies that, for all $p^k \in \mathcal{B}(p^\star, \varepsilon_1)$, [Geiger and Kanzow 2002, Equation 5.77 – Eq. 5.78]

$$-[h_i^k]_j < [\mu_i^k]_j \qquad \forall j \in \mathcal{A}_i(z_i^\star) \quad \forall i \in \mathcal{S}$$

$$[\mu_i^k]_j < -[h_i^k]_j \quad \forall j \in \mathcal{I}_i(z_i^\star) \quad \forall i \in \mathcal{S}.$$

Hence, the block rows of F corresponding to the inequality constraints are locally differentiable and the respective rows in the left-hand side of the inexact Newton stopping criterion (3.6) read

$$\begin{aligned} -[h_i^k]_j - [\nabla h_i^{k\top}(z_i^{k+1} - z_i^k)]_j \quad \forall j \in \mathcal{A}(z_i^\star) \quad \forall i \in \mathcal{S}, \\ [\mu_i^{k+1}]_j \quad \forall j \in \mathcal{I}(z_i^\star) \quad \forall i \in \mathcal{S}. \end{aligned}$$

These block rows are zero because of (3.9) and hence

$$\left\| F^k + \nabla F^{k\top}(p^{k+1} - p^k) \right\| = \left\| \tilde{F}^k + \nabla \tilde{F}^{k\top}(p^{k+1} - p^k) \right\|.$$

Since $\left\|\tilde{F}^k\right\| \leq \left\|F^k\right\|$, we obtain the assertion. This concludes the proof. ■

3.2.2 Inner Convergence

To apply the modified stopping criterion (3.8), we require ADMM to return approximate QP solutions with the correct active set as defined in (3.9). This subsection shows that ADMM locally meets this requirement, first for the iterate $y^{k,l+1}$ obtained by soving the subsystem QPs in Step 7, and then for the averaged iterate $z^{k,l+1}$.

Lemma 3.3 (Active set of the ADMM subsystem QPs)**.** *Suppose Assumption 3.1 holds and let $p^\star$ denote a KKT point of NLP* (3.1) *which satisfies Assumption 3.2. Let $p^k \in \mathcal{B}(p^\star, \varepsilon_1)$ and form QP* (3.2) *with the Hessian $H_i^k = \nabla^2_{z_i z_i} L_i(z_i^k, \nu_i^k, \mu_i^k)$ for all $i \in \mathcal{S}$. Then there exists a unique solution to the two-block QP* (3.3) *$(y^{k,\star}, z^{k,\star}, \nu^{k,\star}, \mu^{k,\star}, \lambda^{k,\star}, \gamma^{k,\star})$ and a constant $\varepsilon_3 > 0$ such that the following holds. If the ADMM initialization $(z^{k,0}, \gamma^{k,0}) \in \mathcal{B}((z^{k,\star}, \gamma^{k,\star}), \varepsilon_3)$, then*

$$\begin{aligned} [h_i^k]_j + [\nabla h_i^{k\top}(y_i^{k,l} - z_i^k)]_j = 0, \quad \forall j \in \mathcal{A}_i(z_i^\star), \quad \forall i \in \mathcal{S}, \quad \forall l > 0, \\ [\mu_i^{k,l}]_j = 0, \quad \forall j \in \mathcal{I}_i(z_i^\star), \quad \forall i \in \mathcal{S}, \quad \forall l > 0. \end{aligned}$$

□

Proof. We first recall that, a), the solution to the two-block QP (3.3) is unique and inherits the active set and strict complementarity from the KKT point $p^\star$. Then, b), we apply the Basic Sensitivity Theorem (BST, Theorem A.3 in Appendix A.2) and show that the optimal active set of the subsystem QP (3.11) is constant for all $(z^{k,l}, \gamma^{k,l})$ close to $(z^{k,\star}, \gamma^{k,\star})$. Then, c), we invoke the ADMM convergence Theorem A.8 to show that all iterates $(y^{k,l}, \mu^{k,l})$ stay at the correct active set.

a) From the proof of Theorem A.6 in the appendix, we have that the solution $p^{k,\star}$ of the partially separable QP (3.2) satisfies LICQ, strict complementarity, and the stronger SOSC condition

$$y_i^\top \nabla^2_{z_i z_i} L_i(z_i^k, \nu_i^k, \mu_i^k) y_i > 0 \quad \text{for all} \quad y_i \neq 0 \quad \text{with} \quad \nabla g_i^{k\top} y_i = 0 \tag{3.10}$$

for all $p^k \in \mathcal{B}(p^\star, \varepsilon_1)$. The reformulation into two-block form preserves (3.10) and LICQ. Moreover, we have that if $z^{k,\star}$ is the minimizer to the single-block QP (A.8), then $y^{k,\star} = z^{k,\star}$ is the

minimizer of the two-block QP (3.3). Thus, strict complementarity also applies to the two-block QP solution.

b) Step 7 solves the subsystem QPs, which in centralized form read

$$\min_{y \in \mathbb{Z}^k} \frac{1}{2} y^\top H^k y + \nabla f^{k\top} y + \gamma^{k,l\top}(y - z^{k,l}) + \frac{\rho}{2} \left\| y - z^{k,l} \right\|_2^2 . \tag{3.11}$$

The centralized subsystem QP is parameterized by $z^{k,l}$ and $\gamma^{k,l}$. Comparing the KKT conditions of the subsystem QP (3.11) and the two-block QP (3.3), we see that if $(z^{k,l}, \gamma^{k,l}) = (z^{k,\star}, \gamma^{k,\star})$, then the solution of (3.11) is given by $(y^{k,\star}, \nu^{k,\star}, \mu^{k,\star})$. Because of a), $(y^{k,\star}, \mu^{k,\star})$ satisfies strict complementarity, the stronger SOSC (3.10), and LICQ. Hence, we can apply the BST Theorem A.3 to the subsystem QP (3.11) perturbed by $(z^{k,l}, \gamma^{k,l}) = (z^{k,\star}, \gamma^{k,\star}) + \xi^{k,l}$. Thus, there exists an $\varepsilon_4 > 0$ such that the active set stays constant if $(z^{k,l}, \gamma^{k,l}) \in \mathcal{B}((z^{k,\star}, \gamma^{k,\star}), \varepsilon_4)$. That is, if $p^k \in \mathcal{B}(p^\star, \varepsilon_1)$ and if $(z^{k,l}, \gamma^{k,l}) \in \mathcal{B}((z^{k,\star}, \gamma^{k,\star}), \varepsilon_4)$, then

$$\begin{aligned} [h_i^k]_j + [\nabla h_i^{k\top}(y_i^{k,l+1} - z_i^k)]_j = 0, \ \forall j \in \mathcal{A}_i(z_i^\star), \quad \forall i \in \mathcal{S}, \\ [\mu_i^{k,l+1}]_j = 0, \ \forall j \in \mathcal{I}_i(z_i^\star), \quad \forall i \in \mathcal{S}. \end{aligned}$$

c) From a) we have that $y_i^\top \nabla^2_{z_i z_i} L_i(z_i^k, \nu_i^k, \mu_i^k) y_i > 0$ for all $y_i \neq 0$ with $\nabla g_i^{k\top} y_i = 0$. Lemma A.1 thus guarantees that the subsystem QP cost functions $f_i^{\mathrm{QP},k}$ are strictly convex over $\mathbb{Z}_i^k$ for all $i \in \mathcal{S}$. Hence, Theorem A.8 guarantees the convergence of ADMM and from (A.24) we obtain

$$\rho \left\| z^{k,l+1} - z^{k,\star} \right\|_2^2 + \frac{1}{\rho} \left\| \gamma^{k,l+1} - \gamma^{k,\star} \right\|_2^2 \leq a_1^2 \left(\rho \left\| z^{k,l} - z^{k,\star} \right\|_2^2 + \frac{1}{\rho} \left\| \gamma^{k,l} - \gamma^{k,\star} \right\|_2^2 \right)$$

with $a_1 < 1$. Since $0 < \rho < \infty$, the above inequality implies that there exists an $\varepsilon_3 > 0$ such that, for all $l \geq 0$, $(z^{k,l}, \gamma^{k,l}) \in \mathcal{B}((z^{k,\star}, \gamma^{k,\star}), \varepsilon_4)$ if $(z^{k,0}, \gamma^{k,0}) \in \mathcal{B}((z^{k,\star}, \gamma^{k,\star}), \varepsilon_3)$ and hence

$$\begin{aligned} [h_i^k]_j + [\nabla h_i^{k\top}(y_i^{k,l} - z_i^k)]_j = 0, \quad \forall j \in \mathcal{A}_i(z_i^\star), \quad \forall i \in \mathcal{S}, \quad \forall l > 0, \\ [\mu_i^{k,l}]_j = 0, \quad \forall j \in \mathcal{I}_i(z_i^\star), \quad \forall i \in \mathcal{S}, \quad \forall l > 0. \end{aligned}$$

This finishes the proof. ■

Lemma 3.3 guarantees that the solution of the subsystem QP stays at the optimal active set if p^k is close to $p^\star$ and if the ADMM initialization is sufficiently good. However, to invoke the modified stopping criterion, we require the averaged iterate z to be at the correct active set. To this end, we make the following assumption which states that no inequality constraints involve variables that are directly coupled between subsystems. This assumption can be met without loss of generality by an appropriate problem reformulation, cf. Subsection 3.3 below.

Assumption 3.3 (Decoupled inequality constraints and consensus problem)**.** *The inequality constraints and the coupling constraint in NLP* (3.1) *satisfy* $\nabla h(z)^\top E^\top = 0$ *for all* $z \in \mathbb{R}^n$*. Furthermore, the right-hand side of the coupling constraint* (3.1d) *satisfies* $b = 0$*.* □

Lemma 3.4 (Active set of the averaged ADMM iterate)**.** *Let Assumption 3.3 and the assumptions of Lemma 3.3 hold and take ε_3 from Lemma 3.3. Furthermore, let the Lagrange multiplier initialization satisfy $M_{\text{avg}}\gamma^{k,0} = 0$, where M_{avg} is the ADMM averaging matrix $M_{\text{avg}} = (I - E^\top(EE^\top)^{-1}E)$. If $p^k \in \mathcal{B}(p^\star, \varepsilon_1)$ and if $(z^{k,0}, \gamma^{k,0}) \in \mathcal{B}((z^{k,\star}, \gamma^{k,\star}), \varepsilon_3)$, then*

$$\begin{aligned}[h_i^k]_j + [\nabla h_i^{k\top}(z_i^{k,l} - z_i^k)]_j = 0,\ \forall j \in \mathcal{A}_i(z_i^\star), \quad \forall i \in \mathcal{S}, \quad \forall l > 0,\\ [\mu_i^{k,l}]_j = 0,\ \forall j \in \mathcal{I}_i(z_i^\star), \quad \forall i \in \mathcal{S}, \quad \forall l > 0.\end{aligned}$$

□

Proof. Lemma 3.3 states that

$$\begin{aligned}[h_i^k]_j + [\nabla h_i^{k\top}(y_i^{k,l} - z_i^k)]_j = 0,\ \forall j \in \mathcal{A}_i(z_i^\star), \quad \forall i \in \mathcal{S}, \quad \forall l > 0,\\ [\mu_i^l]_j = 0,\ \forall j \in \mathcal{I}_i(z_i^\star), \quad \forall i \in \mathcal{S}, \quad \forall l > 0.\end{aligned}$$

From Lemma 2.1 on the ADMM averaging we have

$$z^{k,l+1} = (I - E^\top(EE^\top)^{-1}E)y^{k,l+1}.$$

Hence we obtain

$$\begin{aligned}h^k + \nabla h^{k\top}(z^{k,l+1} - z^k) &= h^k + \nabla h^{k\top}(y^{k,l+1} - z^k) + \nabla h^{k\top}(z^{k,l+1} - y^{k,l+1})\\ &= h^k + \nabla h^{k\top}(y^{k,l+1} - z^k) + \nabla h^{k\top}((I - E^\top(EE^\top)^{-1}E)y^{k,l+1} - y^{k,l+1})\\ &= h^k + \nabla h^{k\top}(y^{k,l+1} - z^k) - \underbrace{\nabla h^{k\top}E^\top}_{=0}(EE^\top)^{-1}Ey^{k,l+1}\\ &= h^k + \nabla h^{k\top}(y^{k,l+1} - z^k)\end{aligned}$$

and thus

$$[h_i^k]_j + [\nabla h_i^{k\top}(z_i^{k,l} - z_i^k)]_j = [h_i^k]_j + [\nabla h_i^{k\top}(y_i^{k,l} - z_i^k)]_j = 0,\ \forall j \in \mathcal{A}_i(z_i^\star), \quad \forall i \in \mathcal{S}, \quad \forall l > 0.$$

■

3.2.3 Local Convergence of Decentralized SQP

Equipped with the convergence guarantees for the outer and inner iterations, we are now ready to state the main result of this section.

Theorem 3.1 (Local convergence of dSQP)**.** *Suppose Assumptions 3.1 and 3.3 hold and let $p^\star$ denote a KKT point of NLP* (3.1) *which satisfies Assumption 3.2. Form QP* (3.2) *with the Hessian $H_i^k = \nabla^2_{z_i z_i} L_i(z_i^k, \nu_i^k, \mu_i^k)$ for all $i \in \mathcal{S}$. Then there exist constants $\varepsilon > 0$ and $\eta \in (0, 1)$ such that the following holds for all $p^0 \in \mathcal{B}(p^\star, \varepsilon)$:*

i) If $\eta^k \leq \eta$ for all $k \geq 0$, then the sequence $\{p^k\}$ generated by Algorithm 3.1 converges to $p^\star$ and the convergence rate is q-linear in the outer iterations.

ii) If additionally $\lim_{k\to\infty} \eta^k = 0$, then the convergence rate is q-superlinear in the outer iterations.

iii) If additionally $\eta^k = O\left(\left\|\tilde{F}^k\right\|\right)$, then the convergence rate is q-quadratic in the outer iterations.

□

Proof. We first show that, a), the ADMM initialization $(z^{k,0}, \gamma^{k,0}) = (z^k, \gamma^k)$ lies inside a neighborhood of the QP solution $(z^{k,\star}, \gamma^{k,\star})$. We then show that, b), this neighborhood can be chosen such that the ADMM iterates remain at the correct active set. Finally, c), we invoke Lemmas 3.1–3.2 to prove convergence.

a) We first choose $\varepsilon \in (0, \varepsilon_2]$ with ε_2 from Lemma 3.1. The exact SQP convergence implies that if $p^k \in \mathcal{B}(p^\star, \varepsilon)$, then $p^{k,\star} \in \mathcal{B}(p^\star, \varepsilon)$ and thus

$$\left\|p^k - p^{k,\star}\right\| = \left\|p^k - p^\star - (p^{k,\star} - p^\star)\right\| \leq \left\|p^k - p^\star\right\| + \left\|p^{k,\star} - p^\star\right\| \leq 2\varepsilon.$$

Recall that $\|x\| \leq \|(x, y)\|$ for any two vectors x and y. Hence,

$$\begin{aligned}
&\left\|(z^k, \lambda^k) - (z^{k,\star}, \lambda^{k,\star})\right\| \leq \left\|p^k - p^{k,\star}\right\| \leq 2\varepsilon \\
\implies &\left\|\begin{bmatrix} I & 0 \\ 0 & E^\top \end{bmatrix}\right\| \left\|(z^k, \lambda^k) - (z^{k,\star}, \lambda^{k,\star})\right\| \leq \left\|\begin{bmatrix} I & 0 \\ 0 & E^\top \end{bmatrix}\right\| \cdot 2\varepsilon \\
\implies &\left\|\begin{bmatrix} I & 0 \\ 0 & E^\top \end{bmatrix}\begin{bmatrix} z^k - z^{k,\star} \\ \lambda^k - \lambda^{k,\star} \end{bmatrix}\right\| \leq \underbrace{\max\left\{1, \left\|E^\top\right\|\right\} \cdot 2\varepsilon}_{\doteq\, \varepsilon_5} \\
\implies &\left\|(z^k, \gamma^k) - (z^{k,\star}, \gamma^{k,\star})\right\| \leq \varepsilon_5.
\end{aligned}$$

b) The dSQP initialization $\gamma_i^0 = E_i^\top \lambda^0$ for all $i \in \mathcal{S}$ implies $M_{\text{avg}}\gamma^0 = 0$ for any $\lambda^0 \in \mathbb{R}^{n_c}$. Since $b = 0$ because of Assumption 3.3, Lemma 2.1 ii) thus implies $M_{\text{avg}}\gamma^{k,l} = 0$ for all $k, l \geq 0$. Lemma 3.4 hence guarantees that the ADMM iterates $(z^{k,l+1}, \gamma^{k,l+1})$ stay at the correct active set, if $(z^{k,0}, \gamma^{k,0}) \in \mathcal{B}((z^{k,\star}, \gamma^{k,\star}), \varepsilon_3)$. If $\varepsilon \leq \min(\varepsilon_2, \varepsilon_3/(2\max\{1, \|E^\top\|\}))$, then $\varepsilon_5 \leq \varepsilon_3$ and thus the ADMM iterates $(z^{k,l+1}, \gamma^{k,l+1})$ stay at the correct active set for all $l \geq 0$ and for all $k \geq 0$.

c) Since the ADMM iterates stay at the correct active set, we can apply the modified stopping criterion (3.8) to guarantee convergence. Moreover, the ADMM convergence Theorem A.8 guarantees that, for all $k \geq 0$, the ADMM iterates satisfy (3.8) for sufficiently large l. By Lemma 3.2, satisfaction of (3.8) together with the correct active set implies satisfaction of the inexact Newton stopping criterion (3.6). Lemma 3.1 then implies q-convergence in the outer iterations. The q-quadratic convergence in iii) follows because $\left\|\tilde{F}^k\right\| \leq \left\|F^k\right\|$ and thus $\eta^k = O\left(\left\|\tilde{F}^k\right\|\right) \implies \eta^k = O\left(\left\|F^k\right\|\right)$. This finishes the proof. ■

Remark 3.3 (Difference to [Stomberg et al. 2022b])**.** *Theorem 3.1 and the preceding convergence analysis on the outer and inner levels are largely based on [Stomberg et al. 2022b]. However, part c) in the proof of [Stomberg et al. 2022b, Lemma 4] fails to address effects of the ADMM averaging step on the active set in the presence of coupled inequality constraints.*[4] *Thus, we have added Assumption 3.3 and adjusted the proof of Lemma 3.4.* □

3.3 Numerical Example: ADMM Active Set

In the previous analysis, Assumption 3.3 ensures ADMM returns an iterate z^{k+1} with the correct active set. Specifically, the assumption states that there exist no coupled inequalities such that the averaging step preserves the active set of the iterate $y^{k,l}$ found by solving the subsystem QPs. To illustrate potential pitfalls regarding the active set of ADMM, we consider the strongly convex QP

$$\min_{x\in\mathbb{R}} 10(x-10)^2 + (x-1)^2 \quad \text{subject to} \quad x \leq 1 \mid \mu. \tag{3.12}$$

The global minimum lies at $x^\star = 1$ with corresponding Lagrange multiplier $\mu^\star = 180$. We first reformulate QP (3.12) as a partially separable QP

$$\begin{aligned} \min_{z_1\in\mathbb{R},z_2\in\mathbb{R}} \; & 10(z_1-10)^2 + (z_2-1)^2 \\ \text{subject to} \quad & z_1 \leq 1 \mid \mu_1 \\ & z_1 - z_2 = 0 \mid \lambda \end{aligned}$$

and then as the two-block QP

$$\min_{(y_1,y_2,z_1,z_2)\in\mathbb{R}^4} 10(y_1-10)^2 + (y_2-1)^2 \tag{3.13a}$$

$$\text{subject to } y_1 \leq 1 \mid \mu_1 \tag{3.13b}$$

$$z_1 - z_2 = 0 \mid \lambda \tag{3.13c}$$

$$y_1 - z_1 = 0 \mid \gamma_1 \tag{3.13d}$$

$$y_2 - z_2 = 0 \mid \gamma_2. \tag{3.13e}$$

That is, we split the objective among two subsystems and assign the inequality constraint to the first subsystem. The unique KKT point of QP (3.13) is given by $y_1^\star = y_2^\star = z_1^\star = z_2^\star = 1$, $\lambda^\star = \gamma_1^\star = \gamma_2^\star = 0$, and $\mu^\star = 180$. Since $E_1 = 1$ and $E_2 = -1$, the averaging matrix is given by

$$M_{\text{avg}} = I - E^\top (EE^\top)^{-1} E = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$$

and the ADMM iterations read

$$y_1^{l+1} = \operatorname*{argmin}_{y_1} \left(10(y_1-10)^2 + \gamma_1^l y_1 + \frac{\rho}{2}(y_1 - z_1^l)^2 \text{ subject to } y_1 \leq 1 \right)$$

[4] A corrected version of the originally published conference paper is available at `https://arxiv.org/abs/2204.08786`.

$$
\begin{aligned}
y_2^{l+1} &= \operatorname*{argmin}_{y_2}\left((y_2-1)^2+\gamma_2^l y_2+\frac{\rho}{2}(y_2-z_2^l)^2\right)=\frac{2-\gamma_2+\rho z_2}{\rho+2}\\
\begin{bmatrix}z_1^{l+1}\\ z_2^{l+1}\end{bmatrix} &= M_{\text{avg}}\left(\begin{bmatrix}y_1^{l+1}\\ y_2^{l+1}\end{bmatrix}+\begin{bmatrix}\gamma_1^l/\rho\\ \gamma_2^l/\rho\end{bmatrix}\right)\\
\gamma_1^{l+1} &= \gamma_1^l+\rho(y_1^{l+1}-z_1^{l+1}),\quad \gamma_2^{l+1}=\gamma_2^l+\rho(y_2^{l+1}-z_2^{l+1}).
\end{aligned}
$$

The KKT point is a fixed point, because setting $(z_i^l, \gamma_i^l) = (1, 0)$ for $i \in \{1, 2\}$ gives

$$
\begin{aligned}
y_1^{l+1} &= \operatorname*{argmin}_{y_1}\left(10(y_1-10)^2+\frac{\rho}{2}(y_1-1)^2 \text{ subject to } y_1\le 1\right)=1\\
y_2^{l+1} &= \frac{2-0+\rho}{\rho+2}=1\\
z_1^{l+1} &= z_2^{l+1}=\frac{y_1^{l+1}+y_2^{l+1}}{2}=1\\
\gamma_1^{l+1} &= \gamma_2^{l+1}=0+\rho(1-1)=0.
\end{aligned}
$$

However, if we perturb the initialization and instead choose $\gamma_1^l = -\gamma_2^l = \varepsilon$ with $\varepsilon \ll 1$, then

$$
\begin{aligned}
y_1^{l+1} &= \operatorname*{argmin}_{y_1}\left(10(y_1-10)^2+\varepsilon y_1+\frac{\rho}{2}(y_1-1)^2 \text{ subject to } y_1\le 1\right)=1\\
y_2^{l+1} &= \frac{2+\varepsilon+\rho}{\rho+2}\\
z_1^{l+1} &= z_2^{l+1}=\frac{y_1^{l+1}+y_2^{l+1}}{2}=\frac{1}{2}\left(1+\frac{2+\varepsilon+\rho}{\rho+2}\right)\neq 1.
\end{aligned}
$$

The perturbation ε causes y_2^{l+1} and hence the averaged variable z_1^{l+1} to leave the active set, i.e., $z_1^{l+1} \neq 1$. To remove the influence of the perturbed y_2^{l+1} on z_1^{l+1}, we add a decision variable to the first subsystem and we rewrite the problem as

$$
\begin{aligned}
\min_{(y_1,z_1,y_2,z_2)\in\mathbb{R}^6}\ & 10\left(\begin{bmatrix}1 & 0\end{bmatrix}y_1-10\right)^2+(y_2-1)^2 && \text{(3.14a)}\\
\text{subject to } & \begin{bmatrix}1 & 0\end{bmatrix}y_1\le 1 \mid \mu_1 && \text{(3.14b)}\\
& \begin{bmatrix}1 & -1\end{bmatrix}y_1=0 \mid \nu_1 && \text{(3.14c)}\\
& \begin{bmatrix}0 & 1\end{bmatrix}z_1-z_2=0 \mid \lambda && \text{(3.14d)}\\
& y_1-z_1=0 \mid \gamma_1 && \text{(3.14e)}\\
& y_2-z_2=0 \mid \gamma_2. && \text{(3.14f)}
\end{aligned}
$$

Here, the first component of y_1 participates in the inequality constraint (3.14b), but only the second component of y_1 is coupled to subsystem two via (3.14d). The averaging matrix becomes

$$
M_{\text{avg}}=\begin{bmatrix}1 & 0 & 0\\ 0 & 0.5 & 0.5\\ 0 & 0.5 & 0.5\end{bmatrix}
$$

and the perturbed iterations read

$$y_1^{l+1} = \begin{bmatrix} 1 \\ 1 \end{bmatrix},\ y_2^{l+1} = \frac{2+\varepsilon+\rho}{\rho+2},\ z_1^{l+1} = \begin{bmatrix} 1 \\ \frac{1}{2}\left(1 + \frac{2+\varepsilon+\rho}{\rho+2}\right) \end{bmatrix},\ \text{and } z_2^{l+1} = \frac{1}{2}\left(1 + \frac{2+\varepsilon+\rho}{\rho+2}\right).$$

The perturbation in y_1^{l+1} hence only affects the second component of z_1^{l+1} such that $h_1^k + \nabla h_1^k(z_1^{l+1} - z_1^k) = h_1(z_1^{l+1}) = 0$ as required by the stopping criterion (3.8). However, this comes at the cost of an additional decision variable and the additional equality constraint (3.14c), which potentially slows down convergence.

3.4 Two-block Decentralized SQP

As discussed in the previous section, small perturbations in combination with coupled inequality constraints can cause the ADMM iterates $z^{k,l}$ to leave the correct active set such that the stopping criterion (3.8) is no longer applicable. Besides problem reformulations that require additional decision variables, this problem can be addressed by a slight modification of dSQP.

Recall that, in each SQP step, Algorithm 3.1 linearizes NLP (3.1), reformulates the resulting QP into two-block form, and applies ADMM. Moreover, dSQP updates the primal variable in Step 12 as $z^{k+1} = z^{k,l}$ instead of $z^{k+1} = y^{k,l}$. This is because the choice $z^{k+1} = z^{k,l}$ ensures $z^{k+1} \in \mathbb{E}$ such that the corresponding block-rows in $\tilde{F}$ vanish and thus the stopping criterion in Step 6 of Algorithm 3.1 can be evaluated in decentralized fashion.

However, updating the SQP iterate with the averaged decision variable $z^{k,l}$ negatively affects the stability of the active set and hence Assumption 3.3 is made to invoke the modified stopping criterion (3.8) for proving convergence in Theorem 3.1. These drawbacks can be overcome by switching the sequence of steps in the derivation of a two-block QP for ADMM, i.e., by first reformulating NLP (3.1) into two-block form and by then linearizing the two-block NLP. This approach leads us to the two-block dSQP method given in Algorithm 3.2.

Figure 3.2 compares the derivation of the two-block QP for ADMM to the steps of Algorithm 3.1. The different strategies could be referred to as *linearize, then decentralize* for dSQP shown in green and *decentralize, then linearize* for two-block dSQP shown in blue. The key difference compared to Algorithm 3.1 is that two primal variable blocks, y and z, are updated on the outer level. This alleviates the need for Assumption 3.3 to ensure stability of the active set in the

Algorithm 3.2 Two-block decentralized SQP for solving (3.1)

SQP initialization: $k = 0$, $(y_i^0, z_i^0, \nu_i^0, \mu_i^0, \gamma_i^0 = E_i^\top \lambda^0)$ for all $i \in \mathcal{S}$, $\eta^0 < 1$, ϵ
while $\|\bar{F}^k\| \not\leq \epsilon$ **do**
  compute $\nabla f_i(y_i^k), \nabla g_i(y_i^k), \nabla h_i(y_i^k), \bar{H}_i^k$ for all $i \in \mathcal{S}$
  construct $\bar{\tilde{F}}^k$ and $\nabla \bar{\tilde{F}}^k$
  ADMM initialization: $l = 0$, $(z_i^{k,0}, \gamma_i^{k,0}, \nu_i^{k,0}, \mu_i^{k,0}) = (z_i^k, \gamma_i^k, \nu_i^k, \mu_i^k)$ for all $i \in \mathcal{S}$
  while $l = 0$ **or** $\|\bar{\tilde{F}}^k + \nabla \bar{\tilde{F}}^{k\top}(\bar{p}^{k,l} - \bar{p}^k)\| \not\leq \eta^k \|\bar{\tilde{F}}^k\|$ **do**
    $(y_i^{k,l+1}, \nu_i^{k,l+1}, \mu_i^{k,l+1}) \leftarrow \min_{y_i \in \bar{\mathbb{Z}}_i^k} \bar{L}_{\rho,i}^k(y_i, z_i^{k,l}, \gamma_i^{k,l})$ for all $i \in \mathcal{S}$
    $z^{k,l+1} = \operatorname{argmin}_{z \in \mathbb{E}} \bar{L}_\rho^k(y^{k,l+1}, z, \gamma^{k,l})$
    $\gamma_i^{k,l+1} = \gamma_i^{k,l} + \rho(y_i^{k,l+1} - z_i^{k,l+1})$ for all $i \in \mathcal{S}$
    $l \leftarrow l + 1$
  end while
  $(y_i^{k+1}, z_i^{k+1}, \nu_i^{k+1}, \mu_i^{k+1}, \gamma_i^{k+1}) = (y_i^{k,l}, z_i^{k,l}, \nu_i^{k,l}, \mu_i^{k,l}, \gamma_i^{k,l})$ for all $i \in \mathcal{S}$
  choose $\eta^{k+1} \leq \eta^k$
  $k \leftarrow k + 1$
end while
return (y_i^k, z_i^k) for all $i \in \mathcal{S}$

convergence analysis, because the map

$$\bar{F}(\bar{p}) \doteq \begin{bmatrix} \nabla_{y_1} \bar{L}_1(y_1, \nu_1, \mu_1, \gamma_1) \\ -\gamma_1 + E_1^\top \lambda \\ g_1(y_1) \\ \min(-h_1(y_1), \mu_1) \\ y_1 - z_1 \\ \vdots \\ \nabla_{y_S} \bar{L}_S(y_S, \nu_S, \mu_S, \gamma_S) \\ -\gamma_S + E_S^\top \lambda \\ g_S(y_S) \\ \min(-h_S(y_S), \mu_S) \\ y_S - z_S \\ \sum_{i \in \mathcal{S}} E_i z_i - b \end{bmatrix}$$

evaluates the inequalities at y and not at z. The Lagrangian for the two-block NLP is defined as

$$\bar{L} \doteq \sum_{i \in \mathcal{S}} \bar{L}_i - \lambda^\top b \doteq \sum_{i \in \mathcal{S}} (f_i(y_i) + \nu_i^\top g_i(y_i) + \mu_i^\top h_i(y_i) + \gamma_i^\top (y_i - z_i) + \lambda^\top E_i z_i) - \lambda^\top b,$$

where the vector of primal-dual variables is defined as $\bar{p} \doteq (y, z, \nu, \mu, \lambda, \gamma)$. Crucially, the two-block dSQP method evaluates the function derivatives at y_i such that, for all $i \in \mathcal{S}$, $\bar{f}_i^{\mathrm{QP},k}(y_i) \doteq$

$\frac{1}{2}(y_i - y_i^k)\bar{H}_i^k(y_i - y_i^k) + \nabla f_i(y_i^k)^\top(y_i - y_i^k)$ with $\bar{H}_i^k \approx \nabla^2_{y_i y_i}\bar{L}_i(y_i^k, \nu_i^k, \mu_i^k)$,

$$\bar{\mathbb{Z}}_i^k \doteq \left\{ y_i \in \mathbb{R}^{n_i} \;\middle|\; \begin{matrix} g_i(y_i^k) + \nabla g_i(y_i^k)^\top(y_i - y_i^k) = 0 \\ h_i(y_i^k) + \nabla h_i(y_i^k)^\top(y_i - y_i^k) \leq 0 \end{matrix} \right\},$$

and

$$\bar{L}_\rho^k(y, z, \gamma) \doteq \sum_{i\in\mathcal{S}} \bar{L}_{\rho,i}^k \doteq \sum_{i\in\mathcal{S}} \left(\bar{f}_i^{\mathrm{QP},k}(y_i) + \gamma_i^\top(y_i - z_i) + \frac{\rho}{2}\|y_i - z_i\|_2^2 \right).$$

Similarly to (3.8), we exclude the inequality constraints inside the area of constant active set when evaluating the ADMM stopping criterion in Step 6 of Algorithm 3.2 and we define

$$\tilde{\bar{F}}(\bar{p}) \doteq \begin{bmatrix} \nabla_{y_1}\bar{L}_1(y_1, \nu_1, \mu_1, \gamma_1) \\ -\gamma_1 + E_1^\top\lambda \\ g_1(y_1) \\ y_1 - z_1 \\ \vdots \\ \nabla_{y_S}\bar{L}_S(y_S, \nu_S, \mu_S, \gamma_S) \\ -\gamma_S + E_S^\top\lambda \\ g_S(y_S) \\ y_S - z_S \\ \sum_{i\in\mathcal{S}} E_i z_i - b \end{bmatrix}. \tag{3.15}$$

In contrast to (3.5), this includes the consensus constraint residual $y - z$. The components $-\gamma + E^\top\lambda$ and $(\sum_{i\in\mathcal{S}} E_i z_i) - b$ are zero at all iterations by similar arguments as in Lemma 3.2 and they are only included in (3.15) for completeness.

Theorem 3.2 (Local convergence of two-block dSQP)**.** *Suppose Assumption 3.1 holds and let $p^\star$ denote a KKT point of NLP* (3.1) *which satisfies Assumption 3.2. Denote the corresonding KKT point of the two-block NLP by $\bar{p}^\star = (z^\star, z^\star, \nu^\star, \mu^\star, \lambda^\star, E^\top\lambda^\star)$. Form the two-block QP with the exact Hessian $\bar{H}_i^k = \nabla^2_{y_i y_i}\bar{L}_i(y_i^k, \nu_i^k, \mu_i^k)$. Then there exist constants $\varepsilon > 0$ and $\eta \in (0, 1)$ such that the following holds for all $\bar{p}^0 \in \mathcal{B}(\bar{p}^\star, \varepsilon)$:*

i) If $\eta^k \leq \eta$ for all $k \geq 0$, then the sequence $\{\bar{p}^k\}$ generated by Algorithm 3.2 converges to $\bar{p}^\star$ and the convergence rate is q-linear in the outer iterations.

ii) If additionally $\lim_{k\to\infty} \eta^k = 0$, then the convergence rate is q-superlinear in the outer iterations.

iii) If additionally $\eta^k = O\left(\left\|\tilde{\bar{F}}^k\right\|\right)$, then the convergence rate is q-quadratic in the outer iterations.

□

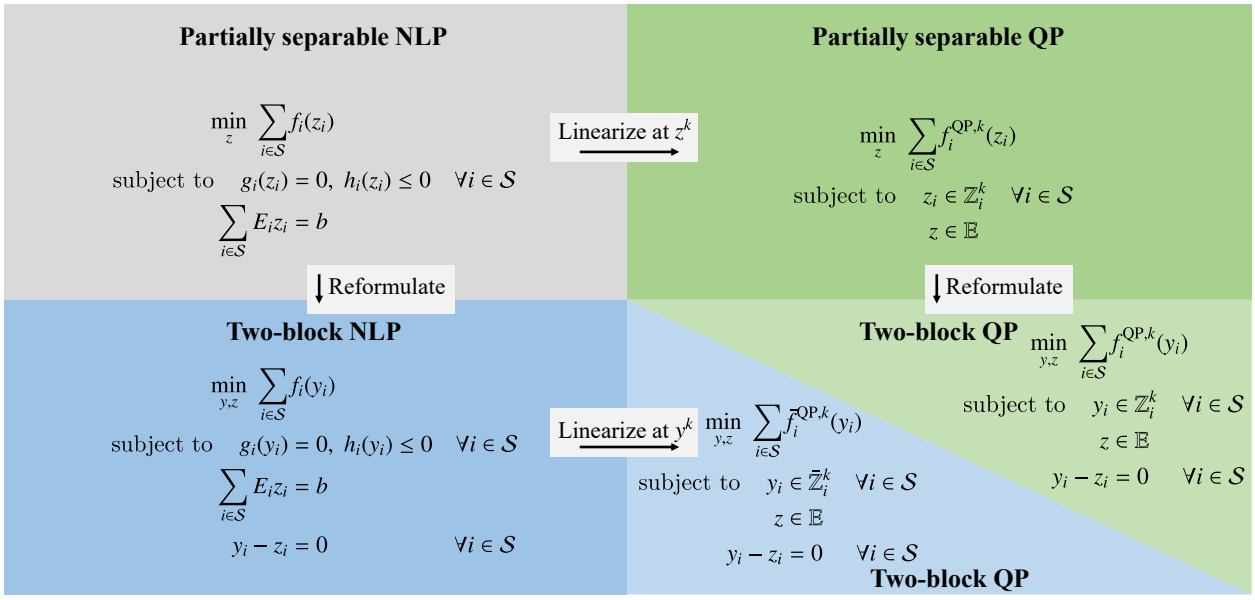


Figure 3.2: Comparison of dSQP (green) and two-block dSQP (blue) showing the transposed steps to derive a two-block QP for ADMM. The two-block QPs use different block variables to evaluate the NLP function derivatives on the outer level. Two-block dSQP evaluates the derivatives at y_i^k to form $\bar{f}_i^{\mathrm{QP},k}$ and $\bar{\mathbb{Z}}_i^k$. In contrast, dSQP forms $f_i^{\mathrm{QP},k}$ and $\mathbb{Z}_i^k$ with the derivatives evaluated at z_i^k.

Proof. The proof proceeds similarly to the convergence analysis in Section 3.2. We first show that, a), the reformulation into two-block form preseves regularity of the KKT point. Then, b), outer convergence based on the inexact-Newton type stopping criterion in Step 6 follows if the iterates (y^k, μ^k) stay at the correct active set. We then show that, c), the ADMM iterates $(y^{k,l+1}, \mu^{k,l+1})$ meet this requirement for all $k, l \ge 0$.

a) Strict complementarity carries over to the two-block NLP, because $y^\star = z^\star$ and hence $h(y^\star) + \mu^\star = h(z^\star) + \mu^\star \neq 0$. LICQ also holds at $\bar{p}^\star$, because the matrix

$$\begin{bmatrix}
\nabla g_1(y_1^\star)^\top & & & & & & \\
& \ddots & & & & & \\
& & \nabla g_S(y_S^\star)^\top & & & & \\
[\nabla h_1(y_1^\star)]^\top_{\mathcal{A}_1(y_1^\star)} & & & & & & \\
& \ddots & & & & & \\
& & [\nabla h_S(y_S^\star)]^\top_{\mathcal{A}_S(y_S^\star)} & & & & \\
I & & & & -I & & \\
& \ddots & & & & \ddots & \\
& & I & & & & -I \\
& & & & E_1 & \dots & E_S
\end{bmatrix}$$

has full row rank. What remains to be shown is that a stricter version of SOSC as in Assumption 3.2 ii) also applies to $\bar{p}^\star$, i.e., we need to show that, for all $i \in \mathcal{S}$,

$$(y_i, z_i)^\top \begin{bmatrix} \nabla^2_{y_i y_i} \bar{L}_i(y_i^\star, \nu_i^\star, \mu_i^\star) & 0 \\ 0 & 0 \end{bmatrix} (y_i, z_i) > 0 \text{ for all } (y_i, z_i) \neq 0 \text{ with} \\ \nabla g_i(y_i^\star)^\top y_i = 0 \text{ and } y_i - z_i = 0. \tag{3.16}$$

By inserting $y_i^\star = z_i^\star$, the last statement is equivalent to

$$z_i^\top \nabla^2_{z_i z_i} L_i(z_i^\star, \nu_i^\star, \mu_i^\star) z_i \quad \text{for all} \quad z_i \neq 0 \quad \text{with} \quad \nabla g_i(z_i^\star)^\top z_i = 0,$$

and thus to Assumption 3.2 ii). Condition (3.16) thus holds.

b) Since Assumption 3.2 carries over to $\bar{p}^\star$, local convergence follows from inexact Newton arguments similar to Lemmas 3.1–3.2, if ADMM terminates at the correct active set, i.e., if

$$[h_i(y_i^k)]_j + [\nabla h_i(y_i^k)^\top (y_i^{k,l} - y_i^k)]_j = 0, \quad \forall j \in \mathcal{A}_i(y_i^\star), \quad \forall i \in \mathcal{S}, \quad \forall l > 0, \tag{3.17a}$$

$$[\mu_i^{k,l}]_j = 0, \quad \forall j \in \mathcal{I}_i(y_i^\star), \quad \forall i \in \mathcal{S}, \quad \forall l > 0. \tag{3.17b}$$

c) As in Lemma 3.3, we apply the BST Theorem A.3 to the perturbed subsystem QPs solved in Step 7 and see that their active sets stay constant, if $(z^{k,l}, \gamma^{k,l}) \approx (z^{k,\star}, \gamma^{k,\star})$. By Lemma A.1, the subsystem QP cost functions $f_i^{\mathrm{QP},k}$ are strictly convex over $\mathbb{Z}_i^k$ and for all $i \in \mathcal{S}$, and we can thus invoke the ADMM convergence guarantees from Theorem A.8 such that (3.17) holds if $(z^{k,0}, \gamma^{k,0}) \in \mathcal{B}((z^{k,\star}, \gamma^{k,\star}), \varepsilon_6)$ for some $\varepsilon_6 > 0$. As discussed in part b) of the proof of Theorem 3.1, this is the case for all $k \geq 0$ if we choose $\varepsilon > 0$ sufficiently small. Hence, the ADMM iterates stay at the correct active set and convergence in the outer iterations follows from b). Q-quadratic convergence in the outer iterations follows, because $\left\|\tilde{\bar{F}}\right\| \leq \left\|\bar{F}\right\|$. This finishes the proof. ■

The crucial difference compared to the proof of Theorem 3.1 is (3.17a), which requires the iterates $y_i^{k,l}$ to be at the correct active set and not $z_i^{k,l}$.

Comparison to the Literature

As mentioned in Chapter 2, there exists a wide range of bi-level decentralized algorithms that combine centralized methods from nonlinear optimization on the outer level with decentralized methods on the inner level. Closely related to the dSQP algorithms presented in this section are schemes combining Sequential Convex Programming (SCP) with Dual Decomposition (DD) [Necoara et al. 2009], SQP and DD [Ma et al. 2011], SCP and ADMM [Le and Nghiem 2020], and SQP and ADMM [Lu 2015; Liu and Jian 2020; Shorinwa and Schwager 2023]. The scheme in [Liu and Tran-Dinh 2020] is guaranteed to converge for problems with non-convex objectives and linear coupling constraints, but it does not allow for non-convex constraints or inequality constraints. In [Lu 2015], also non-convex constraints are considered, but no convergence proof is

presented. The method in [Shorinwa and Schwager 2023], which appeared shortly after our conference paper [Stomberg et al. 2022b], is very similar to dSQP. The main differences to dSQP are that [Shorinwa and Schwager 2023] considers quasi Newton updates for H_i^k and does not terminate ADMM early in each SQP step.

3.5 Numerical Example: Optimal Power Flow

We next analyze the performance of dSQP and two-block dSQP for an AC-OPF example. Motivated by the growing penetration of distributed energy resources in electric power systems, decentralized and distributed optimization and control methods are being developed to augment existing centralized architectures in power system operation, including the solution of OPF problems [Molzahn et al. 2017]. In general, OPF problems seek to minimize power generation costs in electrical grids subject to grid constraints and we refer to [Frank and Rebennack 2016] for more details on OPF formulations and to [Engelmann 2020, Chapter 5] for an overview of distributed OPF.

Here, we consider AC-OPF problems of the form [Engelmann et al. 2019]

$$\min_{\theta,v,P^{\mathrm{g}},Q^{\mathrm{g}}} \sum_{m\in\mathcal{P}} c_{1,m}P_m^2 + c_{2,m}P_m + c_{3,m} \tag{3.18a}$$

subject to

$$P_m^{\mathrm{g}} - P_m^{\mathrm{d}} = v_m \sum_{r\in\mathcal{D}} v_r(G_{mr}\cos(\theta_m-\theta_r) + B_{mr}\sin(\theta_m-\theta_r)) \quad \forall m\in\mathcal{D} \tag{3.18b}$$

$$Q_m^{\mathrm{g}} - Q_m^{\mathrm{d}} = v_m \sum_{r\in\mathcal{D}} v_r(G_{mr}\sin(\theta_m-\theta_r) - B_{mr}\cos(\theta_m-\theta_r)) \quad \forall m\in\mathcal{D} \tag{3.18c}$$

$$P_m^{\mathrm{g}} = Q_m^{\mathrm{g}} = 0 \quad \forall m\in D\setminus\mathcal{P} \tag{3.18d}$$

$$\underline{P}_m \le P_m \le \overline{P}_m,\ \underline{Q}_m \le Q_m \le \overline{Q}_m \quad \forall m\in\mathcal{P} \tag{3.18e}$$

$$\underline{v}_m \le v_m \le \overline{v}_m \quad \forall m\in\mathcal{D} \tag{3.18f}$$

$$v_1 = 1,\ \theta_1 = 0, \tag{3.18g}$$

where $\mathcal{D} = \{1,\dots|D|\}$ is the bus set, $\mathcal{P}\subseteq D$ is the generator set, $Y = G + jB \in \mathbb{C}^{N\times N}$ is the bus admittance matrix, $v = (v_1,\dots,v_N)\in\mathbb{R}^N$ and $\theta = (\theta_1,\dots,\theta_N)\in\mathbb{R}^N$ are the voltage magnitudes and angles, $P^{\mathrm{g}} = (P_1^{\mathrm{g}},\dots,P_N^{\mathrm{g}})\in\mathbb{R}^N$ and $Q^{\mathrm{g}} = (Q_1^{\mathrm{g}},\dots,Q_N^{\mathrm{g}})\in\mathbb{R}^N$ are the injected active and reactive powers, and $P^{\mathrm{d}} = (P_1^{\mathrm{d}},\dots,P_N^{\mathrm{d}})\in\mathbb{R}^N$ are the active and reactive power demands. The objective (3.18a) with coefficients $c_{1,m} > 0$ penalizes power injection and refers to the generation cost. The constraints (3.18b) and (3.18c) are the power flow equations, the inequalities (3.18e)–(3.18f) enforce lower and upper bounds on the injected powers and the voltages, and (3.18g) sets the voltage for the so-called slack bus, chosen here as $m = 1$.

Specifically, we consider the IEEE 118-bus AC-OPF test case which has also served as a benchmark to test distributed OPF based on ADMM, ALADIN, and d-IP [Erseghe 2014; Engelmann

et al. 2019; Engelmann and Faulwasser 2021]. To apply these methods, we use the reformulation as a partially separable NLP (3.1) provided in [Engelmann et al. 2017].[5] Therein, the 118 buses of the test case are partitioned into four subsystems and NLP (3.18) is reformulated as a consensus problem in partially separable form.

The benchmark NLP has $n = 576$ decision variables, $n_g = 470$ nonlinear equality constraints, $n_h = 792$ affine inequality constraints, and $n_c = 52$ coupling constraints. For comparison to the dSQP algorithms, we consider four other optimization methods. The first method for comparison is standalone ADMM, which solves (3.1) directly and which we refer to as ADMM in the remainder of this section. The second and third methods are BL-ALADIN variants, where the coordination QP is solved via essentially d-CG or via an ADMM variant (d-ADMM) [Engelmann et al. 2020]. For all algorithms, we initialize voltage magnitudes as 1 and all other primal-dual variables as 0.

For ADMM, we tune ρ in the set $\{100, 700, 800, 900, 10^3, 10^4\}$ and choose $\rho = 800$ for fastest convergence. For both dSQP variants, we choose $\eta^0 = 0.8$, $\eta^{k+1} = 0.9\eta^k$, tune ρ in the set $\{600, 700, 800\}$, and choose $\rho = 700$. For the BL-ALADIN variants, we use the open-source toolbox ALADIN-α [Engelmann et al. 2021a], tune ρ in the set $\{10, 50, 100, 150, 200, 1000\}$ to obtain $\rho = 150$, and we run 70 inner iterations per ALADIN iteration for ALADIN/d-CG and 100 inner iteration per ALADIN iteration for ALADIN/d-ADMM. We leave the remaining ALADIN parameters at the default values specified by ALADIN-α. For d-IP, we use the same code and parameters as [Engelmann et al. 2021b]. The subsystem NLPs in ADMM and ALADIN are solved with ipopt [Wächter and Biegler 2006] and the subsystem QPs in dSQP are solved with qpOASES [Ferreau et al. 2014]. We use CasADi to compute derivatives in all algorithms [Andersson et al. 2019].

For dSQP and two-block dSQP, we set $H_i^k = \nabla^2_{z_i z_i} L_i^k$ and regularize if needed. To this end, we follow a heuristic regularization procedure and change the sign of negative eigenvalues as follows [Engelmann et al. 2021a]. First, we compute an eigenvalue decomposition $\nabla^2_{z_i z_i} L_i^k = V_i \Lambda_i V_i^\top$. Then, we set $H_i^k = V_i \tilde{\Lambda}_i V_i^\top$, where for all $j \in \{1, \dots, n_i\}$,

$$[\tilde{\Lambda}_i]_{jj} \doteq \begin{cases} \varepsilon, & \text{if } [\Lambda_i]_{jj} \in [-\varepsilon, \varepsilon] \quad \text{with} \quad \varepsilon = 10^{-4}, \\ |\Lambda_{i_{jj}}|, & \text{else,} \end{cases}$$

Figure 3.3 shows the convergence to a local minimizer found by ipopt. For the bi-level algorithms dSQP, two-block dSQP, BL-ALADIN, and d-IP we count the inner iterations. The top plot shows that both dSQP variants perform similarly well compared to ADMM and d-IP. However, the dSQP variants only require to solve QPs on a subsystem level, whereas ADMM solves NLPs. As a result, our prototypical Matlab implementation of ADMM takes 48 s on a desktop computer to achieve $\|z - z^\star\| < 10^{-6}$, whereas dSQP only takes 11 s and two-block dSQP takes 17 s. We note that dSQP and two-block dSQP could likely be further accelerated by switching to sparse solvers for the subsystem QPs. This is because condensing the QPs, which would benefit qpOASES, is

[5] The specific NLP parameterization is available at `https://github.com/alexe15/ALADIN.m/blob/master/test/problem_data/IEEE118busPrbFrm.mat`.

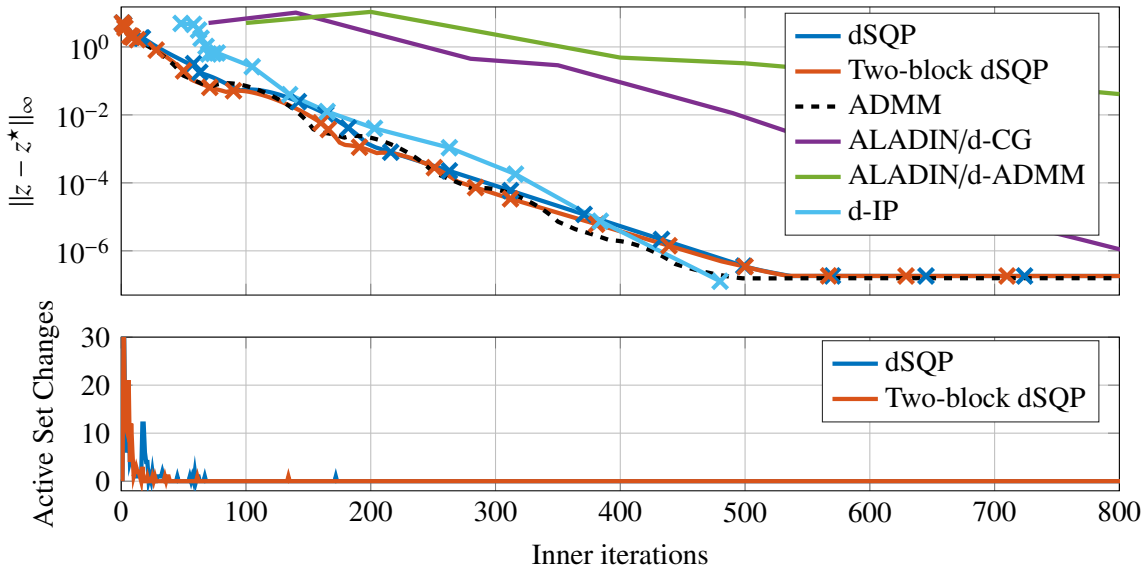


Figure 3.3: Convergence of dSQP, two-block dSQP, ADMM, ALADIN, and d-IP for the 118-bus OPF test case. The crosses for dSQP, two-block dSQP, and d-IP mark outer iterations and the x-axis on the bottom applies to both plots. Figure adapted from [Stomberg et al. 2022b].

typically not favorable in DMPC as it destroys the sparsity of NLP (3.1) which is critical to decentralize computations. Sparse alternatives to qpOASES would be e.g. OSQP or the Proximal Interior-Point Quadratic Programming (PIQP) solver [Schwan et al. 2023].

With respect to communication, all methods require to communicate $2n_c$ floats for all subsystems combined on a neighbor-to-neighbor basis in each inner iteration. The essentially decentralized methods d-IP and ALADIN/d-CG additionally require to communicate two floats globally per inner iteration. Finally, the dSQP variants and d-IP exchange convergence flags among all subsystems when terminating the inner algorithms in each outer iteration.

The bottom plot in Figure 3.3 shows the number of inequality constraints that toggle between being active and inactive for dSQP and two-block dSQP, where a tolerance of 10^{-8} is used to determine whether a constraint is active. Both algorithms settle at a constant active set within the first two hundred iterations. This suggests that the correct active set is reached, as we assume when evaluating the modified inexact Newton stopping criteria.

Remark 3.4 (Relation to bi-level ALADIN, d-ASM, and d-IP)**.** *The bi-level dSQP method takes a similar conceptual approach as BL-ALADIN, d-ASM, and d-IP: Decentralize an established optimization framework by means of an inner algorithm. As in d-IP, convergence under early truncation on the inner level is derived via inexact Newton methods. A distinguishing feature of dSQP to the above algorithms, however, is the inequality constraint treatment on the inner level. Whereas BL-ALADIN, d-ASM, and d-IP manage inequality constraints on the outer level, dSQP solves inequality-constrained QPs on the inner level. As a result, dSQP does not have to explicitly*

identify the correct active set and it does not compute centralized step sizes on the outer level. □

3.6 Summary

This chapter has presented two novel bi-level SQP algorithms where the subproblem QPs are solved in decentralized fashion via ADMM. Local convergence to regular KKT points is guaranteed by tailored inexact Newton-type stopping criteria for the inner level. A small-scale example highlights potential pitfalls for the basic dSQP variant in the presence of coupled inequality constraints, which can be overcome by appropriate problem reformulations or a two-block dSQP reformulation. Both algorithms show competitive performance to further optimization methods for an AC-OPF test case. These results raise the question of whether dSQP can be an interesting algorithmic foundation for nonlinear DMPC. However, the real-time requirements of feedback control benefit from deterministic execution times, i.e., dynamic termination based on stopping criteria is undesirable. Moreover, the impact of non-negligible communication times on closed-loop properties warrants further investigation. We address these challenges in the next chapter.

4 Decentralized Real-Time Iterations

The previous chapter introduced dSQP for solving non-convex partially separable NLPs. This chapter discusses the application of dSQP to nonlinear DMPC and analyzes the stability in closed-loop. To address real-time requirements, we remove the inexact Newton stopping criterion from dSQP and instead apply a fixed number of ADMM iterations in each SQP step. The resulting algorithm can be viewed as an SQP-based RTI scheme where ADMM decentralizes the computations. We therefore first recall RTI schemes and their stability guarantees for centralized NMPC. Then, we prove convergence of dSQP with fixed inner iterations and present the stabilizing dRTI scheme. Finally, we consider an example with coupled inverted pendulums to study the performance for nonlinear open-loop unstable systems. The results of this chapter have appeared in the article [Stomberg et al. 2025a].

4.1 Problem Statement

We consider a centralized nonlinear dynamical system

$$\dot{x}(t_\mathrm{c}) = f^\mathrm{c}(x(t_\mathrm{c}), u(t_\mathrm{c})), \quad x(0) = x_0, \tag{4.1}$$

where $x \in \mathbb{X} \subseteq \mathbb{R}^{n_x}$ is the state, $u \in \mathbb{U} \subseteq \mathbb{R}^{n_u}$ is the input, $t_\mathrm{c} \geq 0$ is time, and $f^\mathrm{c} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$. The sets $\mathbb{X}$ and $\mathbb{U}$ are assumed to be closed. By sampling f^c with a piecewise constant input signal at a sampling interval $\delta > 0$, we obtain the corresponding discrete-time dynamics

$$x(t+1) = x(t) + \int_{t\delta}^{(t+1)\delta} f^\mathrm{c}(x(t_\mathrm{c}), u(t))\, \mathrm{d}t_\mathrm{c} \doteq f^\delta(x(t), u(t)), \quad x(0) = x_0, \tag{4.2}$$

where $f^\delta : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$, $t \in \mathbb{N}_0$ is the discrete time index, and $\mathbb{N}_0$ are the natural numbers extended by zero.

Remark 4.1 (Sparsity of the centralized dynamics)**.** *The careful reader may wonder how the centralized dynamics* (4.2) *relate to the sparse coupling structure present in the cooperative OCP* (1.3)*,*

$$x_i(t+1) = f_i^\delta\left(x_i(t), u_i(t), x_{\mathcal{N}_i^\mathrm{in}}(t)\right), \quad x_i(0) = x_{i,0} \quad \forall i \in \mathcal{S}. \tag{4.3}$$

Indeed, the centralized dynamics f^δ obtained through the integration of the continuous-time dynamics f^c are generally dense and do not admit a sparse decomposition. Nonetheless, the existence of coupled dynamics like (4.3) *with $|\mathcal{N}_i^\mathrm{in}| \ll |\mathcal{S}|$ is a typical assumption in discrete-time cooperative DMPC as it facilitates the use of decentralized optimization [Zeilinger et al. 2013; Conte et al. 2016; Braun and Grüne 2018; Köhler et al. 2019]. Thus, in practice, approximations of the integral in* (4.2) *are required to obtain a sparse cooperative OCP, unless the structure of f^c naturally yields a sparse f^δ. The latter applies for instance to decoupled subsystem dynamics. For the theoretical background to be developed in this chapter, however, we shall assume that f^δ admits an exact and sparse representation like* (4.3) *also for coupled dynamical systems.* □

Throughout this chapter, we consider the setpoint stabilization problem around the origin. That is, we assume $f^{\mathrm{c}}(0,0) = 0$ and we wish to design a centralized feedback law $\kappa_{\mathrm{c}} : \mathbb{R}^{n_x} \to \mathbb{R}^{n_u}$ such that the solutions of (4.2) driven by $u(t) = \kappa_{\mathrm{c}}(x(t))$ satisfy

$$\lim_{t\to\infty} x(t) = 0 \quad \text{and} \quad (x(t), \kappa_{\mathrm{c}}(t)) \in \mathbb{X} \times \mathbb{U} \quad \text{for all} \quad t \in \mathbb{N}_0.$$

To this end, we devise an NMPC scheme where the OCP to be solved at time t reads

$$V(x(t)) \doteq \min_{\boldsymbol{x},\boldsymbol{u}} \sum_{\tau=0}^{N-1} \ell(x[\tau], u[\tau]) + \beta V_{\mathrm{f}}(x[N]) \tag{4.4a}$$

subject to

$$x[\tau+1] = f^{\delta}(x[\tau], u[\tau]) \quad \forall \tau \in \mathbb{I}_{[0,N-1]}, \tag{4.4b}$$

$$x[0] = x(t), \tag{4.4c}$$

$$x[\tau] \in \mathbb{X} \qquad \forall \tau \in \mathbb{I}_{[0,N]}, \tag{4.4d}$$

$$u[\tau] \in \mathbb{U} \qquad \forall \tau \in \mathbb{I}_{[0,N-1]}. \tag{4.4e}$$

Here, $\ell : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^n$ is the stage cost, $V_{\mathrm{f}} : \mathbb{R}^{n_x} \to \mathbb{R}$ is the terminal penalty, and $\beta \geq 1$ is a weighting parameter. We denote by $\boldsymbol{x}^\star \doteq (x^\star[0], \ldots, x^\star[N])$ and $\boldsymbol{u}^\star \doteq (u^\star[0], \ldots, u^\star[N-1])$ a globally optimal solution of OCP (4.4). Let $\mathbb{X}_0 \subseteq \mathbb{R}^{n_x}$ be a closed set with nonempty interior and let OCP (4.4) be feasible for all initial states $x(t) \in \mathbb{X}_0$. We denote the value function of OCP (4.4) by $V : \mathbb{X}_0 \to \mathbb{R}$ and define the centralized NMPC feedback law $\kappa_{\mathrm{c}} : \mathbb{X}_0 \to \mathbb{R}^{n_u}$ as the map from the current state $x(t)$ in (4.4c) to the first part $u^\star[0]$ of the optimal input trajectory.

We next assume that a decomposition of the centralized OCP (4.4) into a set of subsystems $\mathcal{S} = \{1, \ldots, S\}$ is available, cf. [Chanfreut et al. 2021b]. The subsystems can be coupled through the cost (4.4a), dynamics (4.4b), and constraints (4.4d). To reduce notation, we only consider coupling in the state constraints and we note that the extension to coupled input constraints is straight forward. The decomposed version of OCP (4.4) reads

$$\min_{\boldsymbol{x},\boldsymbol{u}} \sum_{i\in\mathcal{S}} J_i\left(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{x}_{\mathcal{N}_i^{\mathrm{in}}}\right) \tag{4.5a}$$

subject to for all $i \in \mathcal{S}$

$$x_i[\tau+1] = f_i^{\delta}\left(x_i[\tau], u_i[\tau], x_{\mathcal{N}_i^{\mathrm{in}}}[\tau]\right) \quad \forall \tau \in \mathbb{I}_{[0,N-1]}, \tag{4.5b}$$

$$x_i[0] = x_i(t), \tag{4.5c}$$

$$x_i[\tau] \in \mathbb{X}_i \qquad \forall \tau \in \mathbb{I}_{[0,N]}, \tag{4.5d}$$

$$u_i[\tau] \in \mathbb{U}_i \qquad \forall \tau \in \mathbb{I}_{[0,N-1]}, \tag{4.5e}$$

$$(x_i[\tau], x_j[\tau]) \in \mathbb{X}_{ij} \qquad \forall j \in \mathcal{N}_i^{\mathrm{in}}, \quad \forall \tau \in \mathbb{I}_{[0,N]}. \tag{4.5f}$$

For all $i \in \mathcal{S}$, let $n_i^{\mathrm{in}} \doteq \sum_{j\in\mathcal{N}_i^{\mathrm{in}}} n_{x_j}$, $\ell_i : \mathbb{R}^{n_{x_i}} \times \mathbb{R}^{n_{u_i}} \times \mathbb{R}^{n_i^{\mathrm{in}}} \to \mathbb{R}$, $V_{\mathrm{f},i} : \mathbb{R}^{n_{x_i}} \to \mathbb{R}$, and

$$J_i\left(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{x}_{\mathcal{N}_i^{\mathrm{in}}}\right) \doteq \sum_{\tau=0}^{N-1} \ell_i\left(x_i[\tau], u_i[\tau], x_{\mathcal{N}_i^{\mathrm{in}}}[\tau]\right) + \beta V_{\mathrm{f},i}(x_i[N]).$$

OCP (4.5) and its centralized counterpart (4.4) are equivalent by design. That is, the components of (4.5) are derived from the components of (4.4) for a given decomposition of the centralized system into a set of subsystems. Considering the centralized system with OCP (4.4) allows to transfer existing RTI stability guarantees from centralized NMPC to distributed NMPC. However, the proposed numerical implementation is decentralized, because we solve (4.5) online via dSQP.

4.2 Centralized Real-Time Iterations

Tailored RTI schemes reduce the computation time when evaluating the control law κ_c in closed-loop [Wolf and Marquardt 2016]. From a numerical point of view, these schemes can be derived from different classes of underlying algorithms. Especially SQP-based schemes have received widespread attention [Li and Biegler 1988; Diehl et al. 2002a; Zavala and Biegler 2009; Gros et al. 2020], but there also exist alternatives based on e.g. indirect optimal control [Ohtsuka 2004; Graichen and Käpernick 2012]. These RTI algorithms warm-start the optimizer with the solution obtained in the previous NMPC step and they apply only few iterations to reduce the computation time.

The stability analysis of RTI schemes is closely related to suboptimal NMPC [Graichen and Kugi 2010] and guarantees for SQP-based RTI algorithms are presented in [Diehl et al. 2003; Diehl et al. 2007]. This subsection recalls centralized NMPC stability guarantees for a generic RTI framework from [Zanelli et al. 2021]. The analysis is applicable to a wide range of RTI schemes and is not limited to a specific optimization algorithm. Instead, the key requirement on the optimizer is local q-linear convergence, a common assumption in the stability analysis of centralized and distributed suboptimal NMPC [Graichen and Kugi 2010; Bestler and Graichen 2019].

To present the RTI framework, we formalize the relation between the system state x, the primal dual-variables p of OCP (4.4), and the NMPC control law κ_c with the following notation. First, $\bar{p} : \mathbb{X}_0 \to \mathbb{R}^{n_p}$ maps from the current state $x(t)$ in (4.4c) to the primal-dual variables $p^\star \in \mathbb{R}^{n_p}$ at a global minimum of OCP (4.4). Second, the matrix $M_{u,p} \in \mathbb{R}^{n_u} \times \mathbb{R}^{n_p}$ with $\|M_{u,p}\| = 1$ selects the control $u^\star[0]$ from $\bar{p}(x)$, i.e., $\kappa_c(x) = u^\star[0] = M_{u,p}\bar{p}(x)$. Third, let the superscript $\cdot^k$ refer to the iteration index of the optimization method which is used to solve OCP (4.4).

The considered RTI scheme proceeds as follows. At time t, the current state $x(t)$ is sampled and the optimization method is initialized with the solution from the previous control step, i.e., $p^0(t) = p^{k_\mathrm{max}}(t-1)$, where $k_\mathrm{max} \in \mathbb{N}$ is the number of optimizer iterations per control step. Then, k_max optimizer iterations are applied to OCP (4.4) and a primal-dual iterate $p^{k_\mathrm{max}}(t)$ is obtained. The control $u(t) = M_{u,p}p^{k_\mathrm{max}}(t)$ is applied to the system and the scheme proceeds with the next sampling step. The optimization method and the system together form the system-optimizer dy-

namics [Zanelli et al. 2021]

$$\begin{bmatrix} x(t+1) \\ p^{k_{\max}}(t+1) \end{bmatrix} = \begin{bmatrix} f^{\delta}(x(t), M_{u,p} p^{k_{\max}}(t)) \\ \Phi(x(t), p^{k_{\max}}(t)) \end{bmatrix}, \tag{4.6}$$

where $\Phi : \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_p}$ maps the OCP initial state $x(t)$ and iterate $p^{k_{\max}}(t)$ to the approximate solution $p^{k_{\max}}(t+1)$ at the next time step. Observe that the system-optimizer dynamics are parameterized by the sampling interval δ. This parameterization is stated explicitly in the top row of the right-hand side of (4.6) and is also included implicitly in our definition of Φ, because $\bar{p}(x(t+1))$ depends on $x(t+1)$ and thus on δ, cf. [Zanelli et al. 2021, Equation 15]. We next summarize the required assumptions and constants to guarantee stability of (4.6) for sufficiently small sampling intervals δ.

Throughout this chapter, let $\|\cdot\|$ denote the Euclidean norm of a vector and the spectral norm of a matrix, respectively, and let $\bar{\mathcal{B}}(a, \varepsilon)$ denote the closed ε neighborhood around a point $a \in \mathbb{R}^n$, $\bar{\mathcal{B}}(a, \varepsilon) \doteq \{b \in \mathbb{R}^n \mid \|a - b\| \leq \varepsilon\}$. Moreover, we denote the Minkowski sum of two sets A and B by $A \oplus B \doteq \{a + b | a \in A, b \in B\}$.

Assumption 4.1 (Value function requirements [Zanelli et al. 2021])**.** *The value function $V : \mathbb{X}_0 \to \mathbb{R}$ of OCP* (4.4) *is continuous and there exist positive constants a_1, a_2, a_3, and $\bar{V}$ such that, for all $x \in \mathbb{X}_{\bar{V}} \doteq \{x \in \mathbb{R}^{n_x} \mid V(x) \leq \bar{V}\}$,*

$$a_1 \|x\|^2 \leq V(x) \leq a_2 \|x\|^2 \tag{4.7a}$$

$$V(f^{\delta}(x, \kappa_{\mathrm{c}}(x))) - V(x) \leq -\delta \cdot a_3 \|x\|^2. \tag{4.7b}$$

Furthermore, there exists a constant $L_{V,x} > 0$ such that, for all $x, x' \in \mathbb{X}_{\bar{V}}$,

$$\left| \sqrt{V(x)} - \sqrt{V(x')} \right| \leq L_{V,x} \|x - x'\|.$$

Moreover, there exists a constant $\hat{r}_x > 0$ such that $\mathbb{X}_{\bar{V}} \oplus \bar{\mathcal{B}}(0, \hat{r}_x) \subseteq \mathbb{X}_0$. □

Assumption 4.2 (Lipschitz controller [Zanelli et al. 2021])**.** *There exists a positive constant $L_{p,x}$ such that, for all $x \in \mathbb{X}_{\bar{V}}$ and for all $x' \in \mathbb{X}_{\bar{V}} \cup \bar{\mathcal{B}}(x, \hat{r}_x)$,*

$$\|\bar{p}(x') - \bar{p}(x)\| \leq L_{p,x} \|x' - x\|.$$

Moreover, $\bar{p}(0) = 0$. □

The assumption $\bar{p}(0) = 0$ holds if the origin is an equilibrium of the system dynamics which minimizes the stage cost ℓ and terminal penalty V_{f}, a common design in stabilizing NMPC. However, the assumption does not hold for *economic NMPC* without further modification. We next adapt [Zanelli et al. 2021, Assumption 10] by assuming, at time t, q-linear optimizer convergence to the current OCP solution $\bar{p}(x(t))$ as follows.

Assumption 4.3 (Q-linear optimizer convergence [Stomberg et al. 2025a])**.** *There exist positive constants $\hat{r}_p > 0$ and $a_p < 1$ such that, for all $x(t) \in \mathbb{X}_{\bar{V}}$ and all $p^0(t) \in \bar{\mathcal{B}}(\bar{p}(x(t)), \hat{r}_p)$, the sequence $\{p^k(t)\}$ of optimizer iterates at time t satisfies*

$$\left\|p^{k+1}(t) - \bar{p}(x(t))\right\| \leq a_p \left\|p^k(t) - \bar{p}(x(t))\right\| \quad \forall k \in \mathbb{N}_0. \qquad \square$$

Assumption 4.4 (Lipschitz continuous-time system dynamics [Zanelli et al. 2021])**.** *The origin is an equilibrium of the centralized system dynamics, $f^{\mathrm{c}}(0,0) = 0$. Moreover, there exist positive finite constants r'_p, ρ, $L^{\mathrm{c}}_{f,x}$, and $L^{\mathrm{c}}_{f,u}$ such that, for all $x', x \in \mathbb{X}_{\bar{V}} \oplus \bar{\mathcal{B}}(0,\rho)$ and for all $u' = M_{u,p}p'$ and $u = M_{u,p}p$ with $p', p \in \bar{\mathcal{B}}(\bar{p}(x), r'_p)$,*

$$\|f^{\mathrm{c}}(x', u') - f^{\mathrm{c}}(x, u)\| \leq L^{\mathrm{c}}_{f,x} \|x' - x\| + L^{\mathrm{c}}_{f,u} \|u' - u\|.$$

Lemma 4.1 (Lipschitz discrete-time dynamics [Zanelli et al. 2021])**.** *Let Assumption 4.4 hold. Then, there exists a positive constant δ_1 such that, if $x \in \mathbb{X}_{\bar{V}}$, $p \in \bar{\mathcal{B}}(\bar{p}(x), r'_p)$, and $\delta \leq \delta_1$, then*

$$\left\|f^{\delta}(x, M_{u,p}p) - x\right\| \leq \delta \cdot \left(L^{\delta_1}_{f,x}\|x\| + L^{\delta_1}_{f,u}\left\|M_{u,p}p\right\|\right),$$

where $L^{\delta_1}_{f,x} \doteq e^{L^{\mathrm{c}}_{f,x}\delta_1} L^{\mathrm{c}}_{f,x}$ and $L^{\delta_1}_{f,u} \doteq e^{L^{\mathrm{c}}_{f,x}\delta_1} L^{\mathrm{c}}_{f,u}$. Furthermore, if $\delta \leq \delta_1$, then

$$\left\|f^{\delta}(x, u') - f^{\delta}(x, u)\right\| \leq \delta L^{\delta_1}_{f,u} \|u' - u\| \tag{4.8}$$

for all $x \in \mathbb{X}_{\bar{V}}$, all $u' = M_{u,p}p'$, $u = M_{u,p}p$ such that $p, p' \in \bar{\mathcal{B}}(r'_p)$. $\square$

Because of the Lyapunov decrease (4.7b), the set $\mathbb{X}_{\bar{V}}$ is positive invariant under the ideal NMPC feedback law κ_{c}. As we show next, this also holds for small perturbations in the controller.

Lemma 4.2 (Forward invariance under perturbed optimizer [Stomberg et al. 2025a])**.** *Let Assumptions 4.1 and 4.4 hold. Then, there exists a positive constant $r''_p \leq r'_p$ such that*

$$f^{\delta}(x, M_{u,p}p) \in \mathbb{X}_{\bar{V}}$$

for all $x \in \mathbb{X}_{\bar{V}}$, all $p \in \bar{\mathcal{B}}(\bar{p}(x), r''_p)$, and if $\delta \leq \delta_1$. $\square$

Proof. For the remainder of this proof, let us define the shorthands

$$x^{\star}_{+} = f^{\delta}(x, M_{u,p}\bar{p}(x)) \quad \text{and} \quad x_{+} = f^{\delta}(x, M_{u,p}p).$$

The Lyapunov decrease (4.7b) in Assumption 4.1 ensures that $V(x^{\star}_{+}) \leq V(x) - \delta a_3 \|x\|^2$. If $x = 0$, then $V(x) = V(x^{\star}_{+}) = 0 < \bar{V}$. Otherwise, if $x \neq 0$, $V(x^{\star}_{+}) \leq \bar{V} - \delta a_3 \|x\|^2 < \bar{V}$. Thus, there exists a positive constant ϵ_V such that

$$V(x^{\star}_{+}) + \epsilon_V \leq \bar{V}.$$

By Assumption 4.1, $\mathbb{X}_{\bar{V}}$ lies in the interior of $\mathbb{X}_0$ and V is continuous over $\mathbb{X}_0$. Thus, there exists a constant $\epsilon_x > 0$ such that [Rawlings et al. 2019, Appendix A.11]

$$|V(x') - V(x^{\star}_{+})| < \epsilon_V \quad \text{for all} \quad \|x' - x^{\star}_{+}\| < \epsilon_x.$$

Let $r_p'' \leq r_p'$. Because the system dynamics are Lipschitz continuous, (4.8) yields, if $\delta \leq \delta_1$,

$$\|x_+ - x_+^\star\| \leq \delta L_{f,u}^{\delta_1} \|p - \bar{p}(x)\| \leq \delta L_{f,u}^{\delta_1} r_p''.$$

Let $\delta \leq \delta_1$ and let $r_p'' < \min\{r_p', \epsilon_x/(\delta L_{f,u}^{\delta_1})\}$. Then, $\|x_+ - x_+^\star\| < \epsilon_x$ such that $V(x_+) < V(x) + \epsilon_V \leq \bar{V}$. By the definition of the level set $\mathbb{X}_{\bar{V}}$, we have that $x_+ \in \mathbb{X}_{\bar{V}}$. This finishes the proof. ■

We next adapt [Zanelli et al. 2021, Lemma 11] to the q-linear convergence Assumption 4.3.

Lemma 4.3 (Contraction [Stomberg et al. 2025a])**.** *Let Assumptions 4.1–4.4 hold and consider the system-optimizer dynamics* (4.6)*. Then, there exist positive constants $r_p \leq \min\{\hat{r}_p, r_p''\}$ and $r_x \leq \hat{r}_x$ such that, for all $x(t) \in \mathbb{X}_{\bar{V}}$ and all $p^{k_{\max}}(t) \in \bar{\mathcal{B}}(\bar{p}(x(t), r_p)$, the following holds. If $\delta \leq \delta_1$ and if $\|x(t+1) - x(t)\| \leq r_x$, then*

$$\left\|p^{k_{\max}}(t+1) - \bar{p}(x(t+1))\right\| \leq a_p \left\|p^{k_{\max}}(t) - \bar{p}(x(t))\right\| + a_p L_{p,x} \|x(t+1) - x(t)\|. \tag{4.9}$$

□

Proof. Under Assumptions 4.1 and 4.4 and because $r_p \leq r_p''$, Lemma 4.2 yields that $x(t+1) = f^\delta(x(t), M_{u,p} p^{k_{\max}}(t)) \in \mathbb{X}_{\bar{V}}$. By Assumption 4.2 on the continuity of the KKT point, we thus have

$$\|\bar{p}(x(t+1)) - \bar{p}(x(t))\| \leq L_{p,x} \|x(t+1) - x(t)\| \leq L_{p,x} r_x.$$

By selecting r_p and r_x such that $r_p \leq \min\{r_p'', \hat{r}_p - L_{p,x} r_x\}$ and $r_x < \min\{\hat{r}_x, \hat{r}_p / L_{p,x}\}$, we obtain

$$\begin{aligned}\left\|p^{k_{\max}}(t) - \bar{p}(x(t+1))\right\| &\leq \left\|p^{k_{\max}}(t) - \bar{p}(x(t))\right\| + \|\bar{p}(x(t+1)) - \bar{p}(x(t))\| \\ &\leq r_p + L_{p,x} r_x \leq \hat{r}_p.\end{aligned}$$

Thus, $\left\|p^{k_{\max}} - \bar{p}(x(t+1))\right\|$ lies inside the q-linear optimizer convergence radius. Recall that we warm-start the solver by setting $p^0(t+1) = p^{k_{\max}}(t)$. Assumptions 4.2 and 4.3 thus yield

$$\begin{aligned}\left\|p^{k_{\max}}(t+1) - \bar{p}(x(t+1))\right\| &\leq (a_p)^{k_{\max}} \left\|p^{k_{\max}}(t) - \bar{p}(x(t+1))\right\| \\ &\leq a_p \left\|p^{k_{\max}}(t) - \bar{p}(x(t+1))\right\| \\ &\leq a_p \left\|p^{k_{\max}}(t) - \bar{p}(x(t))\right\| + a_p \|\bar{p}(t) - \bar{p}(x(t+1))\| \\ &\leq a_p \left\|p^{k_{\max}}(t) - \bar{p}(x(t))\right\| + a_p L_{p,x} \|x(t) - x(t+1)\|\end{aligned}$$

where we have used that $(a_p)^{k_{\max}} \leq a_p < 1$ to obtain the last row. This concludes the proof. ■

To compute a sufficiently small sampling interval $\bar{\delta}$ which guarantees stability of the system-optimizer dynamics, we define the constants

$$\eta \doteq L_{f,x}^{\delta_1} + L_{f,u}^{\delta_1} L_{p,x}, \quad r_{\bar{V}} \doteq \sqrt{\frac{\bar{V}}{a_1}},$$

$$\delta_3 \doteq \min\left(\delta, \delta_1, \frac{r_x}{\eta r_{\bar{V}} + L_{f,u}^{\delta_1} r_p}, \frac{r_p(1-a_p)}{L_{p,x} a_p \left(L_{f,u}^{\delta_1} r_p + \eta r_{\bar{V}}\right)}\right), \quad \kappa \doteq a_p\left(1 + \delta_3 L_{p,x} L_{f,u}^{\delta_1}\right),$$
$$L_V \doteq 2\sqrt{\bar{V}} L_{V,x}, \quad \bar{a} \doteq \frac{a_3}{a_2}, \quad L_e \doteq L_V L_{f,u}^{\delta_1}, \quad L_{V,p} \doteq L_{f,u}^{\mathrm{c}} e^{\delta_1 L_{f,x}^{\mathrm{c}}} L_{V,x}, \quad \tilde{r}_p \doteq \min\left(r_p, \frac{\bar{a}\bar{V}}{L_e}\right)$$
$$\beta' \doteq \frac{\bar{a}\sqrt{a_1}}{4 L_{p,x} a_p \eta}, \quad \delta_4' \doteq \frac{(1-\kappa)\tilde{r}_p \sqrt{a_1}}{\bar{V}^{1/2} L_{p,x} a_p \eta}, \quad \text{and} \quad \delta_5 \doteq \frac{\beta'(1-\kappa)}{L_{V,p}}. \tag{4.10}$$

Lemma 4.4 (Centralized RTI stability [Stomberg et al. 2025a])**.** *Suppose that Assumptions 4.1–4.4 hold and define*

$$\bar{\delta} \doteq \min\left(\delta_3, \delta_4', \delta_5\right). \tag{4.11}$$

If the sampling interval satisfies $\delta \leq \bar{\delta}$, then the origin is a locally exponentially stable equilibrium with region of attraction

$$\Sigma \doteq \left\{\left(x, p^{k_{\max}}\right) \in \mathbb{R}^{n_x+n_p} \;\middle|\; x \in \mathbb{X}_{\bar{V}}, \left\|p^{k_{\max}} - \bar{p}(x)\right\| \leq \tilde{r}_p\right\}$$

for the closed-loop system-optimizer dynamics (4.6). □

Proof. The proof proceeds similarly to the analysis in [Zanelli et al. 2021]. Since the q-linear optimizer convergence Assumption 4.3 differs from [Zanelli et al. 2021, Assumption 10], we replace [Zanelli et al. 2021, Lemma 11] by Lemma 4.3. The optimizer contraction inequality (4.9) is equivalent to [Zanelli et al. 2021, Inequality 26] in the proof of [Zanelli et al. 2021, Proposition 16]. The remaining analysis in [Zanelli et al. 2021] is still applicable such that local exponential stability follows from [Zanelli et al. 2021, Theorem 25]. ■

The above lemma is a specialized version of [Zanelli et al. 2021, Theorem 25]. First, we here only consider quadratic terms $\|x\|^2$ in Assumption 4.1 and the resulting constants in (4.10) whereas [Zanelli et al. 2021, Theorem 25] allows for more general Lyapnuov functions. That is, we set $q = 2$, where q is in the notation of [Zanelli et al. 2021]. Second, we define the optimizer state based on the map $\bar{p}$ from the current system state to the primal-dual variables, because of the dSQP convergence to be presented in the next section. In contrast, [Zanelli et al. 2021, Theorem 25] considers any map $\bar{p}$ as long as the control can be selected as $u = M_{u,p}\bar{p}(x)$. Third, we explicitly allow for $k_{\max} \geq 1$ optimizer iterations per control step to provide more freedom when tuning the bi-level dSQP algorithm. Instead, [Zanelli et al. 2021, Theorem 25] discusses RTI schemes with $k_{\max} = 1$, a common choice in centralized RTI schemes [Diehl et al. 2002a]. We note that the extension to $k_{\max} > 1$ is straight forward, because the q-linear convergence Assumption 4.3 implies an appropriate decrease in the optimizer error also holds for $k_{\max} > 1$. That is, if $k_{\max} = 1$ is stabilizing, so is any $k_{\max} > 1$. Furthermore, Lemma 4.4 and [Zanelli et al. 2021, Theorem 25] differ in the sampling intervals for which stability is guaranteed. Here, we first select a candidate sampling interval δ and design OCP (4.4) such that the OCP value function meets Assumption 4.1. Subsequently, we let Assumptions 4.2–4.4 hold, compute $\bar{\delta}$ via (4.11) and then obtain closed-loop stability guarantees, if our candidate sampling interval δ lies below $\bar{\delta}$. On the other hand, the

analysis in [Zanelli et al. 2021] lets an assumption similar to Assumption 4.1 hold for a range of sampling intervals to then infer closed-loop stability for all sampling intervals below $\bar{\delta}$, not only for a single candidate sampling interval.

Remark 4.2 (Stability guarantees and compensation of computational delay)**.** *Throughout this chapter, we neglect optimizer computation times and instead assume instantaneous feedback. That is, we assume that, at time t, the state measurement $x(t)$ is instantly available, the optimizer iterations are completed without any delay, and the control input $u(t)$ is immediately applied to the plant. In practice, each of these steps will incur inevitable computational delay and, as we will see in Chapter 5, decentralized optimization in DMPC can take up a siginificant portion of the control sampling interval δ due to communication latencies. A classic approach for compensating computational delay in MPC is to, at time t, predict the next state $x(t+1)$ and to solve OCP* (4.4) *with initial condition $x(t+1)$ in* (4.4c) *[Findeisen 2006; Gros et al. 2020]. Notably, the stability guarantees of Lemma 4.4 also hold if such a delay compensation is applied, if a perfect prediction of $x(t+1)$ is available, cf. [Zanelli et al. 2021, Remark 8]. We here omit a detailed analysis for the sake of brevity and note that this would require to define $u(t) \doteq M_{u,p} p^0(t)$, replace the inequality in the q-linear optimizer convergence in Assumption 4.3 by*

$$\left\| p^{k+1}(t) - \bar{p}(x(t+1)) \right\| \leq a_p \left\| p^k(t) - \bar{p}(x(t+1)) \right\|,$$

and to adapt the system-optimizer dynamics (4.6) *and the optimizer contraction Lemma 4.3 accordingly. For the remainder of this chapter and for simplicity, however, we shall assume that instantaneus feedback is available, $u(t) \doteq M_{u,p} p^{k_{\max}}(t)$, and that the system-optimizer dynamics and q-linear optimizer convergence are as stated in* (4.6) *and Assumption 4.3, respectively.* □

4.3 dSQP Convergence with Fixed ADMM Iterations

Lemma 4.4 guarantees closed-loop stability of RTI schemes for sufficiently small sampling intervals, if the optimization method locally converges to a globally optimal OCP solution $\bar{p}(x)$ at a q-linear rate. In this section, we show that dSQP meets this requirement. Chapter 3 discusses the local convergence of the dSQP Algorithm 3.1, where an inexact Newton-type stopping criterion on the inner level terminates ADMM. However, such a stopping criterion is undesirable in control applications, where real-time requirements typically only allow for a fixed number of iterations. Thus, we adapt dSQP as summarized in Algorithms 4.1 and 4.2, where $k_{\max}, l_{\max} \in \mathbb{N}$ are constant. The notation in Step 3 of Algorithm 4.1 denotes that each subsystem $i \in \mathcal{S}$ updates the primal variable y_i as well as the Lagrange multipliers v_i and μ_i to the subsystem constraints (3.2b) and (3.2c), respectively.

Algorithm 4.1 ADMM for solving QP (3.2) in the k–th SQP iteration [Stomberg et al. 2025a]

Initialization: $z_i^{k,0}, \gamma_i^{k,0}, l_{\max}$ for all $i \in \mathcal{S}$
for $l = 0, \dots, l_{\max} - 1$ **do**
  $\left(y_i^{k,l+1}, \nu_i^{k,l+1}, \mu_i^{k,l+1}\right) \leftarrow \min_{y_i \in \mathbb{Z}_i^k} L_{\rho,i}^k\left(y_i, z_i^{k,l}, \gamma_i^{k,l}\right)$ for all $i \in \mathcal{S}$
  $z^{k,l+1} = \operatorname{argmin}_{z \in \mathbb{B}} L_\rho^k\left(y^{k,l+1}, z, \gamma^{k,l}\right)$
  $\gamma_i^{k,l+1} = \gamma_i^{k,l} + \rho\left(y_i^{k,l+1} - z_i^{k,l+1}\right)$ for all $i \in \mathcal{S}$
end for
return $z_i^{k,l_{\max}}$, $\nu_i^{k,l_{\max}}$, $\mu_i^{k,l_{\max}}$, and $\gamma_i^{k,l_{\max}}$ for all $i \in \mathcal{S}$

Algorithm 4.2 dSQP with fixed iterations for solving NLP (3.1) [Stomberg et al. 2025a]

Initialization: $z_i^0, \nu_i^0, \mu_i^0, \gamma_i^0 = E_i^\top \lambda^0, k_{\max}, l_{\max}$ for all $i \in \mathcal{S}$,
for $k = 0, \dots, k_{\max} - 1$ **do**
  compute ∇f_i^k, g_i^k, ∇g_i^k, h_i^k, ∇h_i^k, and H_i^k for all $i \in \mathcal{S}$ and build QP (3.2)
  initialize ADMM in Algorithm 4.1 with z_i^k, γ_i^k for all $i \in \mathcal{S}$ and $l_{\max}$
  run Algorithm 4.1 and denote the output by z_i^{k+1}, ν_i^{k+1}, μ_i^{k+1}, and γ_i^{k+1} for all $i \in \mathcal{S}$
end for
return $z_i^{k_{\max}}$, $\nu_i^{k_{\max}}$, $\mu_i^{k_{\max}}$, and $\gamma_i^{k_{\max}}$ for all $i \in \mathcal{S}$

4.3.1 Outer Convergence

The dSQP convergence analysis in Chapter 3 is based on inexact Newton steps. Because Algorithm 4.1, however, terminates without any stopping criterion, similar Newton-type arguments are not directly applicable. Instead, we present an analysis based on the truncated SQP method in [Zanelli 2021, Algorithm 2].

Lemma 4.5 (Inexact SQP convergence [Stomberg et al. 2025a])**.** *Suppose Assumption 3.1 holds and let $p^\star$ denote a KKT point of NLP* (3.1) *which satisfies the regularity Assumption 3.2. Form QP* (3.2) *at a primal dual point $p^k = (z^k, \nu^k, \mu^k, \lambda^k)$ with the exact Hessian $H_i^k = \nabla^2_{z_i z_i} L_i(z_i^k, \nu_i^k, \mu_i^k)$ for all $i \in \mathcal{S}$ and denote the unique KKT point of QP* (3.2) *by $p^{k,\star}$. Consider an inexact SQP scheme, whose iterates $\left\{p^k\right\}$, for all $k \in \mathbb{N}_0$ and for some $a < 1$, satisfy*

$$\left\|p^{k+1} - p^{k,\star}\right\| \leq a \left\|p^k - p^{k,\star}\right\|. \tag{4.12}$$

Moreover, let $\bar{a}_p$ be a constant such that $\bar{a}_p \in (a, 1)$. Then, there exists a constant $\varepsilon_7 > 0$ such that the following holds. If $p^0 \in \bar{\mathcal{B}}(p^\star, \varepsilon_7)$, then the sequence $\left\{p^k\right\}$ generated by the inexact SQP scheme converges q-linearly to $p^\star$,

$$\left\|p^{k+1} - p^\star\right\| \leq \bar{a}_p \left\|p^k - p^\star\right\| \quad \textit{for all} \quad k \in \mathbb{N}_0. \qquad \square$$

Proof. The centralized SQP convergence Theorem A.6 implies that there exists a constant $\varepsilon_1 > 0$ such that, if $p^0 \in \bar{\mathcal{B}}(p^\star, \varepsilon_1)$, then QP (3.2) is feasible, $p^{k,\star}$ is unique, and

$$\left\|p^{k,\star} - p^\star\right\| \leq C \left\|p^k - p^\star\right\|^2 \tag{4.13}$$

for some $C > 0$ and for all $k \in \mathbb{N}_0$. Let $\varepsilon_7 \leq \varepsilon_1$. Combining the truncated SQP condition (4.12) with the q-quadratic convergence (4.13) yields

$$\begin{aligned}
\left\|p^{k+1} - p^\star\right\| &\leq \left\|p^{k+1} - p^{k,\star}\right\| + \left\|p^{k,\star} - p^\star\right\| \\
&\leq a\left\|p^k - p^{k,\star}\right\| + \left\|p^{k,\star} - p^\star\right\| \\
&\leq a\left(\left\|p^k - p^\star\right\| + \left\|p^{k,\star} - p^\star\right\|\right) + \left\|p^{k,\star} - p^\star\right\| \\
&= a\left\|p^k - p^\star\right\| + (1+a)\left\|p^{k,\star} - p^\star\right\| \\
&\leq a\left\|p^k - p^\star\right\| + (1+a)C\left\|p^k - p^\star\right\|^2 \\
&= \underbrace{\left(a + (1+a)C\left\|p^k - p^\star\right\|\right)}_{\overset{!}{<} \bar{a}_p}\left\|p^k - p^\star\right\|.
\end{aligned}$$

Choosing $\left\|p^0 - p^\star\right\| < (\bar{a}_p - a)/((1+a)C)$ with $a < \bar{a}_p < 1$, we obtain

$$\left\|p^{k+1} - p^\star\right\| \leq \bar{a}_p\left\|p^k - p^\star\right\| \quad \text{for all} \quad k \in \mathbb{N}_0.$$

Thus, if $p^0 \in \bar{\mathcal{B}}(p^\star, \varepsilon_7)$ with $\varepsilon_7 \leq \min(\varepsilon_1, (\bar{a}_p - a)/(1+a)C)$, then the sequence $\left\{p^k\right\}$ converges q-linearly to $p^\star$. ■

Lemma 4.5 guarantees local convergence, if the SQP iterates satisfy condition (4.12). ADMM meets this requirement if the iteration number $l_{\max}$ is chosen appropriately, as we show next.

4.3.2 Inner Convergence

To obtain a sufficient bound on the ADMM iteration number, we first derive an expression for $l_{\max}$ as a function of constants associated with QP (3.2) and the penalty parameter ρ. Subsequently, we discuss the computation of the required constants for computing $l_{\max}$ inside a local convergence neighborhood. To this end, we first define

$$c_1 \doteq \max(1, \|E^\top\|/\rho), \quad D_1 \doteq \begin{bmatrix} M_{\text{avg}} \\ (EE^\top)^{-1}E\rho \end{bmatrix}\begin{bmatrix} I & I \end{bmatrix}, \quad d_1 \doteq \|D_1\|, \quad \text{and} \quad c_2 \doteq d_1 + d_1 d_2 + d_2, \tag{4.14}$$

where $M_{\text{avg}} = I - E^\top(EE^\top)^{-1}E$ is the ADMM averaging matrix and $d_2 > 0$.

Lemma 4.6 (ADMM convergence with fixed iterations [Stomberg et al. 2025a])**.** *Suppose Assumption 3.1 holds and let $p^\star$ denote a KKT point of NLP* (3.1) *which satisfies the regularity Assumption 3.2. Form QP* (3.2) *at a primal-dual point $p^k = (z^k, \nu^k, \mu^k, \lambda^k)$ using the exact Hessian $H_i^k = \nabla^2_{z_i z_i} L_i(z_i^k, \nu_i^k, \mu_i^k)$ for all $i \in \mathcal{S}$. Initialize Algorithm* (4.1) *with $z_i^{k,0} = z_i^k$ and $\gamma_i^{k,0} = E_i^\top \lambda^k$ for all $i \in \mathcal{S}$ and $l_{\max}$. Then, there exist an ADMM contraction factor $0 < a_w < 1$ and constants $d_2, \varepsilon > 0$ such that the following holds for all $0 < a < 1$.*

If $p^k \in \bar{\mathcal{B}}(p^\star, \varepsilon)$ and if

$$l_{\max} \geq 1 + \max\left(0, \left\lceil \log_{a_w}\left(\frac{a}{c_1 c_2}\right)\right\rceil\right) \tag{4.15}$$

with constants c_1 and c_2 defined in (4.14)*, then the iterates $p^{k,l_{\max}} = (z^{k,l_{\max}}, \nu^{k,l_{\max}}, \mu^{k,l_{\max}}, \lambda^{k,l_{\max}})$ returned by Algorithm 4.1 satisfy*

$$\left\|p^{k,l_{\max}} - p^{k,\star}\right\| \leq a \left\|p^{k} - p^{k,\star}\right\|.$$

Furthermore, the active set stays constant, i.e., for all $i \in \mathcal{S}$

$$\left[h_i^k + \nabla h_i^{k\top}(y_i^{k,l} - z_i^k)\right]_{\mathcal{A}_i(z_i^\star)} = 0 \quad \textit{for all} \quad l \in \mathbb{N},$$
$$\left[\mu_i^{k,l}\right]_{\mathcal{I}_i(z_i^\star)} = 0 \quad \textit{for all} \quad l \in \mathbb{N}. \qquad \square$$

Proof. The proof is similar to the proof of Lemma 3.3 and proceeds in five steps. First, a), we recall that $p^{k,\star}$ is regular if $p^k \approx p^\star$. Then, b), we recall that QP (3.2) is strictly convex over the constraints and thus ADMM converges q-linearly in the vector $w \doteq (z, \gamma/\rho)$. Then, c), we recall that the active set stays constant and bound the error of the subsystem QP Step 3. Then, d), we bound the error of the ADMM averaging Step 4. Finally, e), we derive the iteration bound (4.15).

a) Let $0 < \varepsilon \leq \varepsilon_1$, where $\varepsilon_1 > 0$ is the exact SQP convergence radius from Theorem A.6. As discussed in part a) of the proof of Theorem A.6 in the appendix, we can apply the BST Theorem A.3 to QP (3.2) and obtain that $p^{k,\star}$ is a regular KKT point of QP (3.2) if $p^k \in \bar{\mathcal{B}}(p^\star, \varepsilon)$ for some $\varepsilon > 0$. That is, if $p^k \in \bar{\mathcal{B}}(p^\star, \varepsilon)$, then $p^{k,\star}$ satisfies strict complementarity, LICQ, and the stronger SOSC condition

$$y^\top H^k y > 0 \quad \text{for all} \quad y \neq 0 \quad \text{with} \quad \nabla g^{k\top} y = 0. \tag{4.16}$$

b) Thus, Lemma A.1 implies that the QP cost functions $f_i^{\mathrm{QP},k}$ are strictly convex over $\mathbb{Z}_i^k$ for all $i \in \mathcal{S}$. By Lemma A.8, ADMM therefore converges q-linearly in w and, for all $p^k \in \bar{\mathcal{B}}(p^\star, \varepsilon)$ and for some $a_w < 1$,

$$\left\|w^{k,l+1} - w^{k,\star}\right\| \leq a_w \left\|w^{k,l} - w^{k,\star}\right\| \quad \text{for all} \quad l \in \mathbb{N}_0. \tag{4.17}$$

c) As discussed in part b) of the proof of Lemma 3.3, we can view the subsystem QPs solved in Step 3 of Algorithm 4.1 as parametric QPs with parameter $\rho w^{k,l} = (\rho z^{k,l}, \gamma^{k,l})$. By the regularity of $p^{k,\star}$ from a), we can thus apply the BST Theorem A.3 to the centralized subsystem QP (3.11). As a result, there exists a constant $\varepsilon_8 > 0$ such that the map from $w^{k,l} - w^{k,\star}$ to the subsystem QP solution $(y^{k,l+1}, \nu^{k,l+1}, \mu^{k,l+1})$ is continuously differentiable and the active set is constant for all $w^{k,l} \in \bar{\mathcal{B}}(w^{k,\star}, \varepsilon_8)$. That is, if $w^{k,l} \in \bar{\mathcal{B}}(w^{k,\star}, \varepsilon_8)$, then

$$\left[h_i^k + \nabla h_i^{k\top}\left(y_i^{k,l+1} - z_i^k\right)\right]_{\mathcal{A}_i(z_i^\star)} = 0 \quad \text{for all} \quad i \in \mathcal{S},$$
$$\left[\mu_i^{l+1}\right]_{\mathcal{I}_i(z_i^\star)} = 0 \quad \text{for all} \quad i \in \mathcal{S}.$$

Since local continuous differentiability implies local Lipschitz continuity, there exists a constant $d_2 < \infty$ such that, for all $w^{k,l} \in \bar{\mathcal{B}}(w^{k,\star}, \varepsilon_8)$ and for all $p^k \in \bar{\mathcal{B}}(p^\star, \varepsilon)$,

$$\left\|\begin{bmatrix} y^{k,l+1} \\ \nu^{k,l+1} \\ \mu^{k,l+1} \end{bmatrix} - \begin{bmatrix} y^{k,\star} \\ \nu^{k,\star} \\ \mu^{k,\star} \end{bmatrix}\right\| \leq d_2 \left\|w^{k,l} - w^{k,\star}\right\|. \tag{4.18}$$

Finally, the q-linear ADMM convergence (4.17) implies that there exists a constant $\varepsilon \in (0, \varepsilon_1)$ such that $w^{k,l} \in \bar{\mathcal{B}}(w^{k,\star}, \varepsilon_7)$ for all $l \in \mathbb{N}_0$, if $p^k \in \bar{\mathcal{B}}(p^\star, \varepsilon)$.

d) We define

$$\Delta y^{k,l} \doteq y^{k,l} - y^{k,\star}, \quad \Delta z^{k,l} \doteq z^{k,l} - z^{k,\star}, \quad \Delta \nu^{k,l} \doteq \nu^{k,l} - \nu^{k,\star}, \quad \Delta \mu^{k,l} \doteq \mu^{k,l} - \mu^{k,\star}$$
$$\Delta \lambda^{k,l} \doteq \lambda^{k,l} - \lambda^{k,\star}, \quad \text{and} \quad \Delta \gamma^{k,l} \doteq \gamma^{k,l} - \gamma^{k,\star}.$$

The KKT conditions of the averaging Step 4 in Algorithm 4.1 yield, for all $l \in \mathbb{N}_0$,

$$\begin{aligned} z^{k,l+1} &= M_{\text{avg}} \left(y^{k,l+1} + \gamma^{k,l}/\rho \right) + E^\top \left(EE^\top \right)^{-1} b, \\ \lambda^{k,l+1} &= \left(EE^\top \right)^{-1} E\rho \left(y^{k,l+1} + \gamma^{k,l}/\rho \right) - \left(EE^\top \right)^{-1} \rho b. \end{aligned}$$

Likewise,

$$\begin{aligned} z^{k,\star} &= M_{\text{avg}} \left(y^{k,\star} + \gamma^{k,\star}/\rho \right) + E^\top \left(EE^\top \right)^{-1} b, \\ \lambda^{k,\star} &= \left(EE^\top \right)^{-1} E\rho \left(y^{k,\star} + \gamma^{k,\star}/\rho \right) - \left(EE^\top \right)^{-1} \rho b. \end{aligned}$$

and hence

$$\begin{bmatrix} \Delta z^{k,l+1} \\ \Delta \lambda^{k,l+1} \end{bmatrix} = \underbrace{\begin{bmatrix} M_{\text{avg}} \\ \left(EE^\top \right)^{-1} E\rho \end{bmatrix} \begin{bmatrix} I & I \end{bmatrix}}_{D_1} \begin{bmatrix} \Delta y^{k,l+1} \\ \Delta \gamma^{k,l}/\rho \end{bmatrix}. \tag{4.19}$$

Since $d_1 = \|D_1\|$, we obtain

$$\left\| \begin{bmatrix} \Delta z^{k,l+1} \\ \Delta \lambda^{k,l+1} \end{bmatrix} \right\| \leq d_1 \left\| \begin{bmatrix} \Delta y^{k,l+1} \\ \Delta \gamma^{k,l}/\rho \end{bmatrix} \right\| \quad \text{for all} \quad l \in \mathbb{N}_0. \tag{4.20}$$

e) Combining the error bounds (4.18) and (4.20) yields, for all $p^k \in \bar{\mathcal{B}}(p^\star, \varepsilon)$ and for all $l \in \mathbb{N}_0$,

$$\begin{aligned} \left\| p^{k,l+1} - p^{k,\star} \right\| = \left\| \begin{bmatrix} \Delta z^{k,l+1} \\ \Delta \nu^{k,l+1} \\ \Delta \mu^{k,l+1} \\ \Delta \lambda^{k,l+1} \end{bmatrix} \right\| &\leq \left\| \begin{bmatrix} \Delta \nu^{k,l+1} \\ \Delta \mu^{k,l+1} \end{bmatrix} \right\| + \left\| \begin{bmatrix} \Delta z^{k,l+1} \\ \Delta \lambda^{k,l+1} \end{bmatrix} \right\| \\ &\leq d_2 \left\| w^{k,l} - w^{k,\star} \right\| + d_1 \left\| \begin{bmatrix} \Delta y^{k,l+1} \\ \Delta \gamma^{k,l}/\rho \end{bmatrix} \right\| \\ &\leq d_2 \left\| w^{k,l} - w^{k,\star} \right\| + d_1 \left\| \Delta y^{k,l+1} \right\| + d_1 \left\| \Delta \gamma^{k,l}/\rho \right\| \\ &\leq \underbrace{\left(d_1 + d_1 d_2 + d_2 \right)}_{c_2} \left\| w^{k,l} - w^{k,\star} \right\|. \end{aligned} \tag{4.21}$$

From Lemma 2.1 and the initialization $\gamma^0 = E^\top \lambda^0$, we have $\gamma^{k,l} = E^\top \lambda^{k,l}$ for all $k, l \in \mathbb{N}_0$. Thus, for all $l \in \mathbb{N}_0$,

$$w^{k,l} - w^{k,\star} = \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & 0 & 0 & E^\top/\rho \end{bmatrix} \left(p^{k,l} - p^{k,\star} \right)$$

and hence

$$\left\|w^{k,l} - w^{k,\star}\right\| \leq \underbrace{\left\|\begin{bmatrix} I & 0 & 0 & 0 \\ 0 & 0 & 0 & E^\top/\rho \end{bmatrix}\right\|}_{c_1} \left\|p^{k,l} - p^{k,\star}\right\|. \tag{4.22}$$

The q-linear ADMM convergence (4.17) together with the Lipschitz bounds (4.21) and (4.22) yields, for all $p^k \in \mathbb{B}(p^\star, \varepsilon)$,

$$\begin{aligned} \left\|p^{k+1} - p^{k,\star}\right\| = \left\|p^{k,l_{\max}} - p^{k,\star}\right\| &\leq c_2 \left\|w^{k,l_{\max}-1} - w^{k,\star}\right\| \\ &\leq c_2 (a_w)^{l_{\max}-1} \left\|w^{k,0} - w^{k,\star}\right\| \\ &\leq c_1 c_2 (a_w)^{l_{\max}-1} \left\|p^{k,0} - p^{k,\star}\right\| \\ &= \underbrace{c_1 c_2 (a_w)^{l_{\max}-1}}_{\overset{!}{a}} \left\|p^{k} - p^{k,\star}\right\|. \end{aligned}$$

Rearranging $a = c_1 c_2 (a_w)^{l_{\max}-1}$ yields

$$l_{\max} = 1 + \log_{a_w}\left(\frac{a}{c_1 c_2}\right)$$

and we obtain the iteration bound (4.15), because ADMM is only well-defined if $l_{\max} \geq 1$. ■

The iteration bound (4.15) depends on the ADMM contraction factor a_w in (4.17), the constant d_2 in the subsystem QP error (4.18) for computing c_2, and on the parameter $a \in (0, 1)$. Observe that the penalty parameter ρ enters (4.15) through c_1 and also via c_2 and a_w, as we will show below. The design parameter a can be tuned: Choosing a small results in a larger $l_{\max}$, but also increases the sampling interval $\bar{\delta}$ for which stability is guaranteed via Lemma 4.4. With respect to a_w and d_2, there does not, to the best of our knowledge, yet exist a way to compute these constants explicitly for a given QP and for any $w^{k,l} \in \mathbb{R}^{2n}$. However, we can quantify a_w and d_2 for $w^{k,l} \approx w^{k,\star}$ in the area of constant active set.

To this end, we denote the centralized active and inactive sets as

$$\mathcal{A} \doteq \left\{ j \in \{1, \ldots, n_h\} \;\middle|\; \left[h(z^\star)\right]_j = 0 \right\} \quad \text{and} \quad \mathcal{I} \doteq \left\{ j \in \{1, \ldots, n_h\} \;\middle|\; \left[h(z^\star)\right]_j < 0 \right\}.$$

Furthermore, we introduce the shorthands

$$h^k_{\mathcal{A}} \doteq \left[h(z^k)\right]_{\mathcal{A}}, \quad \nabla h^{k\top}_{\mathcal{A}} \doteq \left[\nabla h(z^k)^\top\right]_{\mathcal{A}}, \quad \mu_{\mathcal{A}} \doteq [\mu]_{\mathcal{A}}, \quad \text{and} \quad \mu_{\mathcal{I}} \doteq [\mu]_{\mathcal{I}}.$$

If $p^k \in \bar{\mathcal{B}}(p^\star, \varepsilon)$ with ε from Lemma 4.6, then, for all $l \in \mathbb{N}_0$, the centralized KKT matrix of the subsystem QPs in Step 3 of ADMM is regular and reads

$$K^k \doteq \begin{bmatrix} H^k + \rho I & \nabla g^k & \nabla h^k_{\mathcal{A}} \\ \nabla g^{k\top} & 0 & 0 \\ \nabla h^{k\top}_{\mathcal{A}} & 0 & 0 \end{bmatrix}.$$

We further define the matrix

$$D^k \doteq \rho \left(K^k\right)^{-1} \begin{bmatrix} I & -I \\ 0 & 0 \\ 0 & 0 \end{bmatrix}. \tag{4.23}$$

Lemma 4.7 (Lipschitz constant of the subsystem QP [Stomberg et al. 2025a])**.** *Suppose Assumption 3.1 holds and let $p^\star$ denote a KKT point of NLP* (3.1) *which satisfies the regularity Assumption 3.2. Form QP* (3.2) *at a primal-dual point $p^k = (z^k, \nu^k, \mu^k, \lambda^k)$ using the exact Hessian $H_i^k = \nabla^2_{z_i z_i} L_i(z_i^k, \nu_i^k, \mu_i^k)$ for all $i \in \mathcal{S}$. Initialize Algorithm* (4.1) *with $z_i^{k,0} = z_i^k$ and $\gamma_i^{k,0} = E_i^\top \lambda^k$ for all $i \in \mathcal{S}$ and $l_{\max}$. Let $p^k \in \bar{\mathcal{B}}(p^\star, \varepsilon)$ with ε from Lemma 4.6.*

Then, the Lipschitz constant d_2 in (4.18) *is given by*

$$d_2 = \max_{p^k \in \bar{\mathcal{B}}(p^\star, \varepsilon)} \left\| D^k \right\|. \tag{4.24}$$

□

Proof. Recall from the proof of Lemma 4.6 that the active sets of NLP (3.1), QP (3.2), and the centralized subsystem QP (3.11) to be solved in Step 3 of Algorithm 4.1 are equivalent, if $p^k \in \bar{\mathcal{B}}(p^\star, \varepsilon)$. Moreover, strict complementarity, LICQ, and the stronger SOSC condition (4.16) hold at $(y^{k,l+1}, \mu^{k,l+1})$ for all $l \in \mathbb{N}_0$. The KKT system of the centralized subsystem QP reads

$$\underbrace{\begin{bmatrix} H^k + \rho I & \nabla g^k & \nabla h^k_{\mathcal{A}} \\ \nabla g^{k\top} & 0 & 0 \\ \nabla h^{k\top}_{\mathcal{A}} & 0 & 0 \end{bmatrix}}_{= K^k} \begin{bmatrix} y^{k,l+1} \\ \nu^{k,l+1} \\ \mu^{k,l+1}_{\mathcal{A}} \end{bmatrix} = \begin{bmatrix} -\nabla f^k - \gamma^{k,l} + \rho z^{k,l} \\ -g^k + \nabla g^{k\top} z^{k,0} \\ -h^k_{\mathcal{A}} + \nabla h^{k\top}_{\mathcal{A}} z^{k,0} \end{bmatrix}.$$

Furthermore, $\mu_{\mathcal{I}}^{k,l+1} = 0$. The KKT matrix K^k is regular for all $p^k \in \bar{\mathcal{B}}(p^\star, \varepsilon)$ because of LICQ and the stronger SOSC condition (4.16), cf. [Nocedal and Wright 2006, Lemma 16.1]. The above KKT system yields

$$\begin{bmatrix} y^{k,l+1} \\ \nu^{k,l+1} \\ \mu^{k,l+1}_{\mathcal{A}} \end{bmatrix} = \left(K^k\right)^{-1} \begin{bmatrix} -\nabla f^k - \gamma^{k,l} + \rho z^{k,l} \\ -g^k + \nabla g^{k\top} z^{k,0} \\ -h^k_{\mathcal{A}} + \nabla h^{k\top}_{\mathcal{A}} z^{k,0} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} y^{k,\star} \\ \nu^{k,\star} \\ \mu^{k,\star}_{\mathcal{A}} \end{bmatrix} = \left(K^k\right)^{-1} \begin{bmatrix} -\nabla f^k - \gamma^{k,\star} + \rho z^{k,\star} \\ -g^k + \nabla g^{k\top} z^{k,0} \\ -h^k_{\mathcal{A}} + \nabla h^{k\top}_{\mathcal{A}} z^{k,0} \end{bmatrix}.$$

Thus,

$$\begin{bmatrix} \Delta y^{k,l+1} \\ \Delta \nu^{k,l+1} \\ \Delta \mu^{k,l+1}_{\mathcal{A}} \end{bmatrix} = \rho \left(K^k\right)^{-1} \begin{bmatrix} I & -I \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \Delta w^{k,l+1}. \tag{4.25}$$

Because $\mu_{\mathcal{I}}^{k,l+1} = \mu_{\mathcal{I}}^{k,\star} = 0$, we obtain (4.18) with $d_2 = \max_{p^k \in \bar{\mathcal{B}}(p^\star, \varepsilon)} \left\| D^k \right\|$. ■

Equipped with (4.24) for computing the constant d_2, the last constant required for evaluating the ADMM iteration bound (4.15) is the contraction factor a_w. To compute a_w, we next show that,

inside $\bar{\mathcal{B}}(p^\star, \varepsilon)$, ADMM is an asymptotically stable LTI system. To this end, we first partition the top block rows of D^k defined in (4.23) as

$$\begin{bmatrix} I & 0 & 0 \end{bmatrix} D^k \doteq \begin{bmatrix} T^k & -T^k \end{bmatrix}, \tag{4.26}$$

which yields the block matrix $T^k \in \mathbb{R}^{n \times n}$. The ADMM system matrix reads

$$A^k \doteq \begin{bmatrix} M_{\text{avg}} T^k & M_{\text{avg}}(I - T^k) \\ (I - M_{\text{avg}})T^k & (I - M_{\text{avg}})(I - T^k)) \end{bmatrix}. \tag{4.27}$$

Lemma 4.8 (ADMM contraction factor [Stomberg et al. 2025a])**.** *Suppose Assumption 3.1 holds and let $p^\star$ denote a KKT point of NLP* (3.1) *which satisfies the regularity Assumption 3.2. Form QP* (3.2) *at a primal-dual point $p^k = (z^k, \nu^k, \mu^k, \lambda^k)$ using the exact Hessian $H_i^k = \nabla^2_{z_i z_i} L_i(z_i^k, \nu_i^k, \mu_i^k)$ for all $i \in \mathcal{S}$. Initialize Algorithm* (4.1) *with $z_i^{k,0} = z_i^k$ and $\gamma_i^{k,0} = E_i^\top \lambda^k$ for all $i \in \mathcal{S}$ and $l_{\max}$. Let $p^k \in \bar{\mathcal{B}}(p^\star, \varepsilon)$ with ε from Lemma 4.6.*

Then, the ADMM iterations read

$$w^{k,l+1} - w^{k,\star} = A^k \left(w^{k,l} - w^{k,\star} \right). \tag{4.28}$$

Furthermore, the ADMM contraction factor $a_w < 1$ in (4.17) *is given by*

$$a_w = \max_{p^k \in \bar{\mathcal{B}}(p^\star, \varepsilon)} \left\| A^k \right\|. \tag{4.29}$$

□

Proof. Lemma 4.6 implies that the active set of the centralized subsystem QP stays constant for all $l \in \mathbb{N}_0$ if $p^k \in \bar{\mathcal{B}}(p^\star, \varepsilon)$. Thus, the subsystem QP error is given by (4.25). Rewriting the top block rows of (4.25) yields

$$\Delta y^{k,l+1} = \begin{bmatrix} T^k & -T^k \end{bmatrix} \Delta w^{k,l}. \tag{4.30}$$

From Lemma 4.6, recall the error of the ADMM averaging step (4.19),

$$\Delta z^{k,l+1} = M_{\text{avg}} \left(\Delta y^{k,l+1} + \Delta \gamma^{k,l} / \rho \right). \tag{4.31}$$

By inserting (4.30) into (4.31), we obtain

$$\Delta z^{k,l+1} = \begin{bmatrix} M_{\text{avg}} T^k & M_{\text{avg}}(I - T^k) \end{bmatrix} \Delta w^{k,l}. \tag{4.32}$$

From the dual update in Step 5 of ADMM, we obtain

$$\begin{aligned} \frac{\Delta \gamma^{k,l+1}}{\rho} &= \frac{\Delta \gamma^{k,l}}{\rho} + \left(\Delta y^{k,l+1} - \Delta z^{k,l+1} \right) \quad \text{and hence} \\ \frac{\Delta \gamma^{k,l+1}}{\rho} &= \begin{bmatrix} (I - M_{\text{avg}})T^k & (I - M_{\text{avg}})(I - T^k) \end{bmatrix} \Delta w^{k,l}. \end{aligned} \tag{4.33}$$

By combining (4.32) and (4.33) with the definition $w = (z, \gamma/\rho)$, we can write ADMM as the LTI system (4.28),

$$\Delta w^{k,l+1} = \underbrace{\begin{bmatrix} M_{\text{avg}} T^k & M_{\text{avg}}(I - T^k) \\ (I - M_{\text{avg}})T^k & (I - M_{\text{avg}})(I - T^k)) \end{bmatrix}}_{A^k} \Delta w^{k,l}. \tag{4.34}$$

Hence,

$$\left\|\Delta w^{k,l+1}\right\| \leq \left\|A^k\right\| \left\|\Delta w^{k,l}\right\|.$$

The q-linear ADMM convergence (4.17) yields, for all $p^k \in \bar{\mathcal{B}}(p^\star, \varepsilon)$,

$$\left\|A^k\right\| = \max_{\Delta w^{k,l} \neq 0} \frac{\left\|A^k \Delta w^{k,l}\right\|}{\left\|\Delta w^{k,l}\right\|} = \max_{\Delta w^{k,l} \neq 0} \frac{\left\|\Delta w^{k,l+1}\right\|}{\left\|\Delta w^{k,l}\right\|} < 1$$

and so we obtain the worst-case contraction factor a_w inside $\bar{\mathcal{B}}(p^\star, \varepsilon)$ via (4.29). ■

Combining Lemmas 4.6–4.8 shows how many inner ADMM iterations $l_{\max}$ suffice to guarantee local convergence of the inexact SQP steps via Lemma 4.5.

4.3.3 Local Convergence of dSQP with Fixed Iterations

We next combine the outer convergence (Lemma 4.5) and inner convergence (Lemma 4.6) to derive the q-linear convergence of Algorithm 4.2, as required by the centralized RTI stability Lemma 4.4.

Theorem 4.1 (dSQP convergence with fixed ADMM iterations [Stomberg et al. 2025a])**.** *Suppose Assumption 3.1 holds and let $p^\star$ denote a KKT point of NLP* (3.1) *which satisfies the regularity Assumption 3.2. Form QP* (3.2) *at a primal-dual point $p^k = (z^k, \nu^k, \mu^k, \lambda^k)$ using the exact Hessian $H_i^k = \nabla^2_{z_i z_i} L_i(z_i^k, \nu_i^k, \mu_i^k)$ for all $i \in \mathcal{S}$. Let $0 < a < \bar{a}_p < 1$. Then, there exists a constant $\varepsilon > 0$ such that the following holds.*

Compute the constants d_2 and a_w via (4.24) *and* (4.29)*. If $p^0 \in \bar{\mathcal{B}}(p^\star, \varepsilon)$ and if the number $l_{\max}$ of ADMM iterations per SQP iteration satisfies* (4.15)*, then the sequence $\{p^k\}$ generated by Algorithm 4.2 converges to $p^\star$ and*

$$\left\|p^{k+1} - p^\star\right\| \leq \bar{a}_p \left\|p^k - p^\star\right\| \quad \textit{for all} \quad k \in \mathbb{N}_0. \qquad \square$$

Proof. The outer convergence (Lemma 4.5) implies that there exists a constant $\varepsilon_7 > 0$ such that the following holds. If $p^0 \in \bar{\mathcal{B}}(p^\star, \varepsilon_7)$ and if

$$\left\|p^{k+1} - p^{k,\star}\right\| \leq a \left\|p^k - p^{k,\star}\right\| \quad \text{for all} \quad k \in \mathbb{N}_0,$$

then the sequence $\{p^k\}$ converges q-linearly to $p^\star$ with contraction factor $\bar{a}_p$. By the inner convergence (Lemma 4.6), there exists a constant $\varepsilon > 0$ such that the iterates produced by ADMM meet this requirement for all $p^k \in \bar{\mathcal{B}}(p^\star, \varepsilon)$. By choosing $\varepsilon \in (0, \varepsilon_7)$, we thus obtain q-linear convergence of $\{p^k\}$ to $p^\star$. This finishes the proof. ■

Algorithm 4.3 Decentralized Real-Time Iterations [Stomberg et al. 2025a]

1: Initialization: dSQP warm start $p_i^0(0)$ for all $i \in \mathcal{S}$, $k_{\max}$, $l_{\max}$
2: **for** NMPC step $t = 0, 1, 2, \ldots$ **do**
3: measure the current system state $x_i(t)$ for all $i \in \mathcal{S}$
4: update the OCP initial condition in (3.1b) with $x_i(t)$ for all $i \in \mathcal{S}$
5: warm start dSQP by setting $p_i^0(t) = p_i^{k_{\max}}(t-1)$ for all $i \in \mathcal{S}$
6: **for** SQP iteration $k = 0, \ldots, k_{\max} - 1$ **do**
7: evaluate ∇f_i^k, g_i^k, ∇g_i^k, h_i^k, ∇h_i^k, and H_i^k for all $i \in \mathcal{S}$
8: initialize ADMM by setting $z_i^{k,0} = z_i^k$ and $\gamma_i^{k,0} = \gamma_i^k$ for all $i \in \mathcal{S}$
9: **for** ADMM iteration $l = 0, \ldots, l_{\max} - 1$ **do**
10: solve subsystem QP $\left(y_i^{k,l+1}, \nu_i^{k,l+1}, \mu_i^{k,l+1}\right) \leftarrow \min_{y_i \in \mathbb{Z}_i^k} L_{\rho,i}^k\left(y_i, z_i^{k,l}, \gamma_i^{k,l}\right)$ for all $i \in \mathcal{S}$
11: solve $z^{k,l+1} = \operatorname{argmin}_{z \in \mathbb{E}} L_\rho^k\left(y^{k,l+1}, z, \gamma^{l,k}\right)$ via decentralized averaging procedure
12: update $\gamma_i^{k,l+1} = \gamma_i^{k,l} + \rho\left(y_i^{k,l+1} - z_i^{k,l+1}\right)$
13: **end for**
14: set $p_i^{k+1} = p_i^{k,l_{\max}}$ for all $i \in \mathcal{S}$
15: **end for**
16: select the control input $u_i(t) = u_i^{k_{\max}}[0]$ from $p_i^{k_{\max}}(t)$ for all $i \in \mathcal{S}$
17: apply the control input $u_i(t)$ for all $i \in \mathcal{S}$
18: **end for**

Similar to the convergence analysis in Chapter 3, the proof of Theorem 4.1 is divided into the convergence on the outer and inner levels. However, in contrast to Chapter 3, Theorem 4.1 does not rely on inexact Newton-type arguments, but instead ensures sufficient accuracy of ADMM via condition (4.12) on the QP error $p^{k+1} - p^{k,\star}$.

4.4 Closed-Loop Stability with Decentralized Real-Time Iterations

This section presents the dRTI scheme and combines the q-linear convergence of dSQP with the centralized RTI stability Lemma 4.4 to guarantee stability of dRTI.

Consider the setpoint stabilization for the network $\mathcal{S}$ of coupled dynamical subsystems with dynamics (4.5b). Algorithm 4.3 summarizes the dRTI scheme, where the dSQP Algorithm 4.2 is used to obtain approximate solutions to OCP (4.5) online. For the sake of readability, we suppress the dependence of the SQP and ADMM iterates on the current time t in Steps 6–15 of Algorithm 4.3. Moreover, we collect the primal-dual variables stored on subsystem $i \in \mathcal{S}$ in the vector $p_i \doteq (z_i, \nu_i, \mu_i, \gamma_i)$.

In each control step, the system state $x(t)$ is measured. Then, the optimizer is warm-started with

the solution obtained in the previous NMPC step, i.e. $p^0(t) \doteq p^{k_{\max}}(t-1)$, and dSQP applies $k_{\max}$ SQP iterations and $l_{\max}$ ADMM iterations per SQP iteration. We define the distributed NMPC control law $\kappa_{\mathrm{d}} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_u}$ as the map from the current centralized state $x(t)$ and optimizer initialization $p^0(t)$ to the first part $u^{k_{\max}}[0]$ of the predicted input trajectory obtained with dSQP. Under the feedback law κ_{d}, the centralized system (4.2) and dSQP together form the system-optimizer dynamics (4.6). Here, Φ maps the current system state $x(t)$ and dSQP iterate $p^{k_{\max}}(t)$ to the dSQP solution at the next time step. We are now ready to state the main result of this chapter.

Theorem 4.2 (dRTI stability [Stomberg et al. 2025a])**.** *Suppose that Assumptions 4.1 and 4.4 hold and that $\bar{p}(0) = 0$. Further assume that, for all $x \in \mathbb{X}_{\bar{V}}$, Assumption 3.1 holds and that $\bar{p}(x)$ satisfies the regularity Assumption 3.2. Consider a constant $a \in (0, 1)$ for Theorem 4.1. For all $x \in \mathbb{X}_{\bar{V}}$, denote the dSQP convergence radius and contraction factor from Theorem 4.1 with $\varepsilon(x)$ and $\bar{a}_p(x)$, compute the constants $d_2(x)$ and $a_w(x)$ according to Lemmas 4.7 and 4.8, and assume the number of ADMM iterations $l_{\max}$ satisfies* (4.15)*. Set $\hat{r}_p = \min_{x \in \mathbb{X}_{\bar{V}}} \varepsilon(x)$ and $a_p = \max_{x \in \mathbb{X}_{\bar{V}}} \bar{a}_p(x)$, and compute $\tilde{r}_p$ and $\bar{\delta}$ via* (4.10)–(4.11)*. Then, the following holds.*

If the control sampling interval satisfies $\delta \leq \bar{\delta}$, then the origin is a locally exponentially stable equilibrium with region of attraction

$$\Sigma \doteq \left\{ \left(x, p^{k_{\max}} \right) \in \mathbb{R}^{n_x + n_p} \;\middle|\; x \in \mathbb{X}_{\bar{V}}, \left\| p^{k_{\max}} - \bar{p}(x) \right\| \leq \tilde{r}_p \right\}$$

for the closed-loop system-optimizer dynamics (4.6)*.* □

Proof. Consider the reformulation of OCP (4.5) as a partially separable NLP (3.1). According to the dSQP convergence (Theorem 4.1), Algorithm 4.3 converges q-linearly to the primal-dual solution $\bar{p}(x(t))$ for any initialization $p^0 \in \bar{\mathcal{B}}(\bar{p}(t), \tilde{r}_p)$ and for all $x \in \mathbb{X}_{\bar{V}}$. Thus, dSQP satisfies the q-linear convergence Assumption 4.3. The KKT points are regular (Assumption 3.2), the functions in NLP (3.1) are twice continuously differentiable, and the Jacobian ∇g^k is continuously differentiable with respect to the initial condition $x(t)$. Thus, the Lipschitz continuity of the map $\bar{p}$ (Assumption 4.2) follows from the BST Theorem A.3. Therefore, the exponential stability of the origin for the closed-loop system-optimizer dynamics follows from Lemma 4.4. ■

Remark 4.3 (Estimated region of attraction)**.** *If the initial state $x(0)$ and the iterate $p^{k_{\max}}(0)$ returned by dSQP in the first NMPC step lie inside Σ, then the system-optimizer dynamics converge to the origin. Recall that $\mathbb{X}_{\bar{V}}$ is the set over which Assumptions 3.1, 3.2, 4.1, and 4.4 hold. The optimizer attraction radius $\tilde{r}_p$ can be computed via* (4.10)*.* □

Remark 4.4 (Continuity of V and Lipschitz continuity of $\sqrt{V}$)**.** *Assumption 4.1 requires the value function V to be continuous for all $\mathbb{X}_{\bar{V}}$ and $\sqrt{V}$ to be Lipschitz continuous at the origin. Both conditions hold, if V is Lipschitz continuous over $\mathbb{X}_{\bar{V}}$ and twice continuously differentiable at the origin [Zanelli et al. 2021]. This is indeed the case under the regularity Assumption 3.2, because the functions in NLP* (3.1) *are twice continuously differentiable in $(z, x(t))$ [Fiacco 1983, Theorem 3.4.1].* □

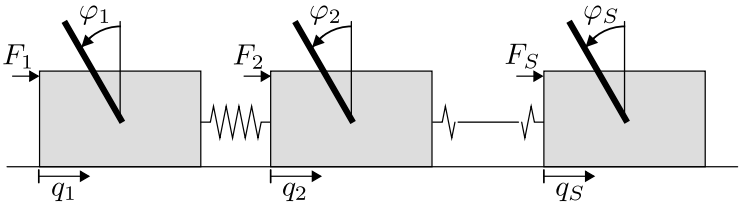


Figure 4.1: Chain of coupled inverted pendulums on carts. Figure adapted from [Stomberg et al. 2025a].

Remark 4.5 (Stability close to global optima)**.** *The centralized RTI stability Lemma 4.4 requires optimizer convergence to a global minimum of OCP* (4.4)*. Consequently, this assumption carries over to the dRTI stability Theorem 4.2. In general, OCP* (4.5) *is non-convex for nonlinear system dynamics and thus dSQP can only be expected to converge to local minima. However, we observe stability in simulations and experiments despite the convergence to local minima. Centralized and decentralized RTI stability guarantees which only require convergence to local minima appear to be so far not known and remain a topic for future work, both for OCPs with and without terminal constraints. For the former, the infeasibility of RTIs challenges standard arguments whereas the latter typically invokes the OCP value function and thus, by definition, only applies to global minima, cf. [Rawlings et al. 2019, Chapter 2].* □

Remark 4.6 (Application to linear-quadratic DMPC)**.** *In the special case where NLP* (4.5) *is a partially separable convex QP, dSQP reduces to ADMM. Due to the q-linear convergence* (4.17) *of ADMM for convex QPs, Theorem 4.2 then guarantees stability for any ADMM iteration number* $l_{\max}$ *per NMPC step, if the sampling interval* $\delta \leq \bar{\delta}$*. This follows by replacing the primal-dual variables p by the vector* $w = (z, \gamma/\rho)$ *in the system-optimizer dynamics* (4.6)*, cf. [Zanelli 2021].* □

4.5 Numerical Example: Coupled Inverted Pendulums

To illustrate the stabilizing behavior of dRTI, we consider a chain of coupled inverted pendulums as shown in Figure 4.1. Each pendulum is attached to a cart and neighboring carts are coupled via springs. The control task is to swing up the pendulums and stabilize the upright equilibrium.

System Model

For each pendulum $i \in \mathcal{S}$, q_i is the cart position, φ_i denotes the angular deviation from the upright position, and $x_i \doteq (q_i, \dot{q}_i, \varphi_i, \dot{\varphi}_i) \in \mathbb{R}^4$ is the state. The carts are actuated and the control input of subsystem i is the force $u_i = F_i \in [-100\,N, 100\,N]$ applied to the cart. Let $M_c = 2\,\text{kg}$ and $m = 0.25\,\text{kg}$ be the masses of each cart and pendulum, respectively, denote the pendulum length

by $l = 0.2\,\mathrm{m}$ and the spring stiffness by $k = 0.1\,\mathrm{N/m}$. For all $i \in \mathcal{S}$, the equations of motion read

$$\ddot{q}_i = \frac{u_i + \frac{3mg}{4}\sin(\varphi_i)\cos(\varphi_i) - \frac{ml}{2}\dot{\varphi}_i^2 \sin(\varphi_i) + F_i^{\text{left}} + F_i^{\text{right}}}{M_\text{c} + m - \frac{3m}{4}(\cos(\varphi_i))^2},$$
$$\ddot{\varphi}_i = \frac{3g}{2l}\sin(\varphi_i) + \frac{3}{2l}\cos(\varphi_i)\ddot{q}_i,$$

where $g = 9.81\,\mathrm{m/s^2}$ is the gravity of earth. If applicable, the coupling forces to the left and right neighbors in the chain are given by $F_i^{\text{left}} = k(q_{i-1} - q_i)$ and $F_i^{\text{right}} = k(q_{i+1} - q_i)$, respectively. The above equations of motion yield the continuous-time dynamics

$$\dot{x}_i(t) = f_i^\text{c}(x_i(t), u_i(t), x_{\mathcal{N}_i^\text{in}}), \quad x_i(0) = x_{i,0} \quad \text{for all} \quad i \in \mathcal{S},$$

where $f_i^\text{c} : \mathbb{R}^4 \times \mathbb{R} \times \mathbb{R}^{n_i^\text{in}} \to \mathbb{R}^4$ and $x_{\mathcal{N}_i}$ are the coupled states from neighboring carts. That is, $x_{\mathcal{N}_i} \doteq q_{i+1}$ if $i = 1$, $x_{\mathcal{N}_i} \doteq q_{i-1}$ if $i = S$, and $x_{\mathcal{N}_i} \doteq (q_{i-1}, q_{i+1})$ otherwise. To obtain the discrete-time dynamics $f_i^h : \mathbb{R}^4 \times \mathbb{R} \times \mathbb{R}^{n_i^\text{in}} \to \mathbb{R}^4$, we discretize the above continuous-time dynamics via the explicit fourth-order Runge-Kutta method (RK4) with shooting interval $h = 40\,\mathrm{ms}$. For this discretization, we keep the control u_i *and* the coupled states $x_{\mathcal{N}_i^\text{in}}$ constant over the shooting interval when evaluating f_i^c. Assuming coupled states to be constant over the shooting interval is a simplification, which preserves the coupling structure. That is, for all $i \in \mathcal{S}$, f_i^h only depends on the immediate neighbors in the chain. However, this simplification worsens the accuracy because the integration order with respect to coupled states reduces to one. If desired, a more accurate discretization can be obtained via distributed multiple shooting [Savorgnan et al. 2011]. Therein, state trajectories are expressed as linear combinations of basis functions and a sparse coupling between subsystems is obtained by matching the basis function coefficients between neighbors.

Optimal Control Problem Design

We design a stabilizing OCP (4.5) such that the value function satisfies inequalities (4.7) from Assumption 4.1 for the control sampling interval $\delta = 40\,\mathrm{ms}$. In order to reduce the computational burden, we opt for an OCP design without terminal constraints as proposed by [Limon et al. 2006]. The only inequalities present in the resulting OCP are input box constraints. Hence, the omission of terminal constraints here avoids state constraints altogether, which usually makes an OCP easier to solve [Rawlings et al. 2019, p. 144]. However, if desired, terminal constraints could be added to the OCP.

For all $i \in \mathcal{S}$, we choose separable quadratic costs

$$\ell_i(x_i, u_i) = \frac{1}{2}x_i^\top Q_i x_i + \frac{1}{2}u_i^\top R_i u_i \quad \text{and} \quad V_{\text{f},i} = \frac{\beta_2}{2}x_i^\top P_i x_i,$$

with $R_i = 0.001$ and with $Q_i = \operatorname{diag}\left(1, 10^{-4}, 10, 10^{-4}\right)$ from [Mills et al. 2009]. The components $\beta_2 \geq 1$ and $P_i \in \mathbb{R}^{4\times4}$ are determined as follows.

First, the coupling between carts is neglected, the dynamics are discretized via RK4 with the sampling interval $\delta = 40\,\mathrm{ms}$, and the discretized dynamics are linearized at the origin. Thus we

obtain, for all $i \in \mathcal{S}$, a linearized model (A_i, B_i) with matrices $A_i \in \mathbb{R}^{4\times4}$ and $B_i \in \mathbb{R}^{4\times1}$. This allows to solve the discrete-time algebraic Riccati equation

$$P_i = A_i^\top P_i A_i - (A_i^\top P_i B_i)(R_i + B_i^\top P_i B_i)^{-1}(B_i^\top P_i A_i) + Q_i \tag{4.35}$$

for each pendulum individually to obtain P_i and a terminal control law

$$u_i = K_i x_i \quad \text{with} \quad K_i = -(B_i^\top P_i B_i + R_i)^{-1}(B_i^\top P_i A_i).$$

Then, we discretize the centralized continuous-time dynamics $f^{\mathrm{c}} : \mathbb{R}^{4S} \times \mathbb{R}^S \to \mathbb{R}^{4S}$ via RK4 with the control sampling interval $\delta = 40\,\mathrm{ms}$ to obtain discretized dynamics $\tilde{f}^\delta : \mathbb{R}^{4S} \times \mathbb{R}^S \to \mathbb{R}^{4S}$. The notation $\tilde{f}^\delta$ highlights that neighboring states are not assumed to be constant in the discretization process, unlike the discretization f_i^h. Then, we linearize $\tilde{f}^\delta$ at the origin to obtain the centralized LTI model (A, B) with dense matrices $A \in \mathbb{R}^{4S\times4S}$ and $B \in \mathbb{R}^{4S\times S}$. Here, the terminal controller $K \doteq \mathrm{diag}(K_1, \dots, K_S)$ found via the above Riccati equation for the decoupled models (A_i, B_i) also stabilizes the coupled centralized system, i.e., the matrix $A_k \doteq A + BK$ is Schur stable. Note that this design approach is not guaranteed to stabilize any coupled linearized system, but is successful for the pendulum example considered here.

Let $Q \doteq \mathrm{diag}(Q_1, \dots, Q_S)$, $R \doteq \mathrm{diag}(R_1, \dots, R_S)$, and $P \doteq \mathrm{diag}(P_1, \dots, P_S)$ denote the centralized weight matrices. The centralized cost functions read

$$\ell(x, u) = \frac{1}{2}x^\top Q x + \frac{1}{2}u^\top R u \quad \text{and} \quad V_{\mathrm{f}} = \frac{\beta_2}{2}x^\top P x.$$

If the centralized terminal penalty V_{f} locally meets the sufficient decrease condition

$$V_{\mathrm{f}}\left(\tilde{f}^\delta(x, Kx)\right) - V_{\mathrm{f}}(x) \leq -\ell(x, Kx), \tag{4.36}$$

then the OCP value function V satisfies inequalities (4.7) for all $\beta \geq 1$ in a neighborhood around the origin [Rawlings et al. 2019, p. 145]. Since A_K is Schur stable, we can choose β_2 such that (4.36) holds in a neighborhood around the origin by slight modifications to the design procedure described in [Rawlings et al. 2019, Section 2.5.5]. To this end, consider the Lyapunov equation

$$A_K^\top \beta_2 P A_K + \mu\tilde{Q} = \beta_2 P, \tag{4.37}$$

where the paramter $\mu > 1$ is in the notation of [Rawlings et al. 2019, Section 2.5.5]. We can solve the Lyapunov equation to obtain $\tilde{Q} = -\beta_2\left(A_K^\top P A_K - P\right)/\mu$. Then, we increase β_2 until the matrix $\Delta Q \doteq \tilde{Q} - Q_K$ is positive definite, where $Q_K \doteq Q + K^\top R K$. We choose $\mu = 1.01$ and obtain $\beta_2 = 1.1$. To see why this ensures that the sufficient decrease condition holds in a neighborhood of the origin, we rewrite (4.36) as

$$V_{\mathrm{f}}\left(\tilde{f}^\delta(x, Kx)\right) - V_{\mathrm{f}}(x) \leq -\frac{1}{2}x^\top Q_k x. \tag{4.38}$$

Because $\tilde{Q} = Q_K + \Delta Q$ with positive definite ΔQ, inequality (4.38) holds, if

$$V_{\mathrm{f}}\left(\tilde{f}^\delta(x, Kx)\right) - V_{\mathrm{f}}(x) \leq -\frac{1}{2}x^\top \tilde{Q} x. \tag{4.39}$$

The Lyapunov equation (4.37) yields

$$V_\mathrm{f}(x) = V_\mathrm{f}(A_K x) + \frac{\mu}{2} x^\top \tilde{Q} x$$

and inserting the above equation into (4.39) gives

$$V_\mathrm{f}\left(\tilde{f}^\delta(x, Kx)\right) - V_\mathrm{f}(A_K x) \leq \frac{\mu - 1}{2} x^\top \tilde{Q} x. \tag{4.40}$$

That is, the sufficient decrease condition (4.36) holds if (4.40) is met. This is indeed the case in a neighborhood around the origin, because the discretized pendulum dynamics are twice continuously differentiable [Rawlings et al. 2019, Section 2.5.5].

To summarize, the OCP is designed by tuning Q and R, computing P via individual Riccati equations for the decoupled pendulum dynamics, and scaling the factor β_2 until V_f meets the sufficient decrease condition. A similar approach of computing terminal costs via decoupled dynamics is suggested in [Stewart et al. 2011, Remark 4]. While this procedure is not guaranteed to stabilize any coupled system, the approach produces an appropriate terminal cost and terminal controller for the considered pendulum example. Further decentralized control designs for linear systems are discussed in the classic textbook [Siljak 1991].

Swing-Up Simulation

The dRTI scheme is tested in simulations for a chain of $S = 20$ pendulums. The prototypical Matlab implementation uses CasADi to evaluate derivatives in the SQP scheme and is available online.[1] The subsystem QPs are solved in each ADMM iteration via OSQP with tolerances $\epsilon_\mathrm{abs} = \epsilon_\mathrm{rel} = \epsilon_\mathrm{dual} = \epsilon_\mathrm{dual} = 10^{-8}$, where the tolerances are defined in [Stellato et al. 2020]. The pendulums are simulated by propagating the centralized system dynamics via RK4 with integration step size equivalent to the sampling interval $\delta = 40\,\mathrm{ms}$. That is, the simulation does not keep neighboring states constant within the integration interval and thus is more accurate than the discretization used in the OCP. We tune the ADMM penalty paramter by choosing $\rho = 1$ from the set $\rho = \{0.1, 1, 10, 100\}$ for fastest convergence to a local minimizer found by ipopt [Wächter and Biegler 2006].

We analyze three different test cases with varying initial conditions and solver settings as summarized in Table 4.1, where GN refers to the Gauss-Newton Hessian approximation $H^k \doteq \nabla^2_{zz} f^k$. In each case, the pendulums initially rest in the lower equilibrium position $x_i(0) = (q_i(0), 0, \pi, 0)$, where the initial displacement $q_i(0)$ varies between test cases. The control task is to stabilize the upright equilibrium $x_i = 0$ for all $i \in \mathcal{S}$. In all simulations, the sampling interval is $\delta = 40\,\mathrm{ms}$, the horizon length is $T = 0.4\,\mathrm{s}$, and the scaling parameter in the OCP cost function is set to $\beta = 1$. OCP (4.5) is rewritten as a partially separable NLP like (3.1) as demonstrated in Example 2.1. Here, we penalize all state copies with a quadratic penaly and weight 10^{-5} such that the regularity Assumption 3.2 holds in a neighborhood around the origin.

[1] `https://github.com/OptCon/real-time-dmpc`

Table 4.1: Test case specifications. All cases consider a sampling interval $\delta = 40\,\text{ms}$ [Stomberg et al. 2025a].

Case	$q_i(0)$	h [ms]	N	H	$k_{\max}$	$l_{\max}$	n	n_g	n_h	n_c
1	-1^i	40	10	exact	1	6	1518	880	440	418
2	i	40	10	exact	3	6	1518	880	440	418
3	i	57	7	GN	2	3	1104	640	320	304

The further parameters in Table 4.1 are the shooting interval h in the OCP, the discrete-time horizon $N = T/h$, the type of Hessian used to build QP (3.2), the number of outer and inner iterations $k_{\max}, l_{\max}$, and the NLP dimensions.

For test cases one and two, we choose the exact Hessian $\nabla^2_{zz} L$ if it is positive definite and the GN Hessian approximation $\nabla^2_{zz} f$ otherwise. For case three, we always select the GN Hessian approximation. In the first NMPC step, dSQP is initialized with a local minimum found by ipopt. Subsequently, we warm start dSQP with the solution produced in the previous NMPC step.

Table 4.2 summarizes the control performance and execution times. To measure performance, we compare the averaged closed-loop cost

$$J_{\text{cl}} \doteq \frac{1}{t_n + 1} \sum_{t=0}^{t_n} \sum_{i \in \mathcal{S}} \ell_i(x_i(t), u_i(t)), \tag{4.41}$$

where $t_n \doteq T_f/\delta$ and where $T_f = 10\,\text{s}$ denotes the simulated time span. Moreover, we measure the dSQP execution time on a desktop computer, where each simulation is completed ten times to account for stochasticity in the time measurements. Specifically, we measure the overall dSQP execution time for all subsystems combined, summarized in columns two and three of Table 4.2. This measurement encompasses all steps in dSQP, including the creation and desctruction of intermediate data structures that would be avoided in embedded implementations. Therefore, columns four and five report time measurements for each subsystem individually, where only essential steps in dSQP are included. The per-subsystem measurements only include code blocks for evaluating derivatives via CasADi, solving the subsystem QPs via OSQP, evaluating the averaging Step 4 of ADMM as an efficient matrix-vector product, and performing the dual update in Step 4 of ADMM. Column six summarizes the percentage of per-subsystem execution times that were faster than the sampling interval δ.

Figure 4.2 shows the closed-loop system-optimizer trajectories for the cart positions q_i, pendulum angles φ_i, control inputs u_i, and primal-dual variables p. The swing up is successful for all test cases and dSQP converges to a local minimum found by ipopt. The initial conditions for the cart discplaments are more challenging in cases two and three than in case one. Therefore, case two requires more outer iterations $k_{\max}$ than case one to achieve the swing up and stabilize the upper equilibrium, which also increases the execution times. Case three improves the solve time compared to case two by choosing the GN Hessian approximation and by increasing the shooting interval h, a well-known technique in RTI schemes for reducing computation times [Diehl et al.

Table 4.2: Computation times per NMPC step and closed-loop performance J_{cl} for a prototypical Matlab implementation. The fast computation times demonstrate the real-time feasibility of decentralized real-time iterations [Stomberg et al. 2025a].

Case	all subsystems combined		per subsystem			J_{cl}
	median [ms]	max. [ms]	median [ms]	max. [ms]	$\leq \delta$	
1	239.07	448.19	12.03	29.24	100 %	65.86
2	697.04	2394.02	35.10	210.38	93.54 %	156.05
3	227.06	746.51	10.76	60.25	99.90 %	180.66

2002a]. This reduces the NLP dimensions and requires less time to form QP (3.2), because the GN Hessian approximation is evaluated offline. However, this worsens the control performance and is not covered by the dRTI stability guarantees, because Theorem 4.2 only holds if $h = \delta$ and if $H = \nabla^2_{zz} L$.

The solve times reported in Table 4.2 are mostly faster than the control sampling interval δ. However, a decentralized implementation for execution on multiple machines would not necessarily be real-time feasible, depending on the hardware setup and the communication latencies. For instance, the experimental results reported in the next chapter indicate that an additional execution time of 10 ms per subsystem can be expected for 6 ADMM iterations, if Ethernet communication is used. Thus, test cases one and three in the pendulum simulations are conceivably real-time feasible in practice, but the settings in test case two would require either a faster implementation or more computation power.

Validity of Assumptions and dRTI Parameter Estimation

We next comment on the assumptions made in the stability Theorem 4.2 and the computation of $l_{\max}$ via (4.15) and $\bar{\delta}$ via (4.11). As discussed above, the OCP value function V satisfies the Lyapunov function requirements (4.7) in Assumption 4.1 because of the terminal penalty. The simulations further show that the KKT points found by ipopt are regular (Assumption 3.2) close to the setpoint. Consequently, the continuity conditions in Assumptions 4.1 and 4.2 hold, albeit for a KKT point $p^\star$ at a local minimum. Finally, the pendulum dynamics satisfy Assumption 4.4 and the functions in NLP (3.1) are three times continuously differentiable (Assumption 3.1).

We estimate the ADMM iteration number and control sampling interval guaranteeing stability for $\delta = 40$ ms via Theorem 4.2 as follows. First, we proceed similarly to [Zanelli et al. 2021] and estimate the constants required for computing δ_5 via simulations where we solve OCP (4.4) with ipopt to evaluate the ideal NMPC feedback law κ_{c}. These simulations yield $a_1 = 0.5326$, $a_2 = 7.1530$, $a_3 = 0.2113$, $L^{\mathrm{c}}_{f,x} = 86.2704$, $L^{\mathrm{c}}_{f,u} = 3.6685$, $L_{p,x} = 51.3647$, $L_{V,x} = 1.7908$, and $L_{V,p} = 207.1090$. From the QP approximation of OCP (4.5) at $x = 0$, we obtain the constants $c_1 = 1.7321$ and $c_2 = 251.5737$ for computing $l_{\max}$ via (4.15). To estimate the ADMM contraction factor, we formulate the ADMM system matrix A_k for the QP approximation at the setpoint. The

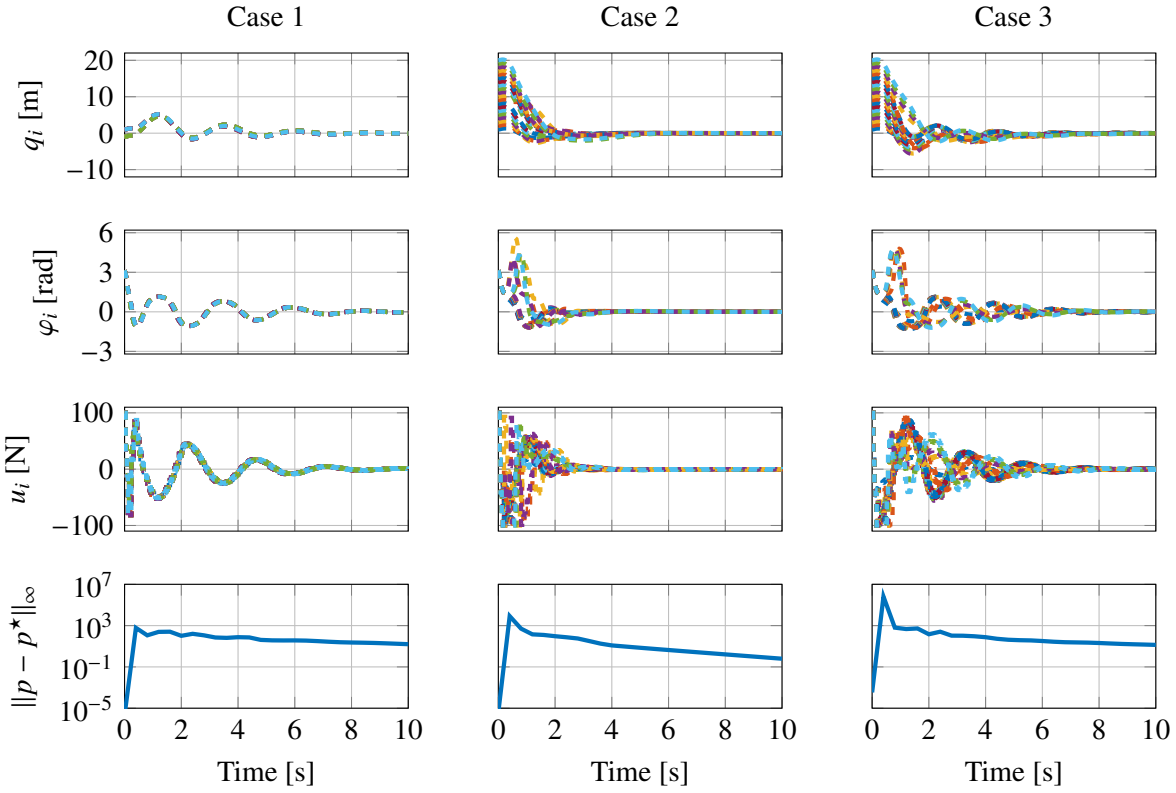


Figure 4.2: Simulation results for the coupled inverted pendulums [Stomberg et al. 2025a]. The y-axis labels on the left apply to all columns and the x-axis labels on the bottom apply to all rows.

singular value $\|A_k\|$ is close to 1. To slightly reduce conservatism, we instead estimate a_w by sampling the ADMM LTI dynamics for random w and obtain $a_w = 0.9989$. Nonetheless, the estimates (4.11) for δ_5 and (4.15) for $l_{\max}$ yield that $l_{\max} = 24,007$ ADMM iterations suffice for stability for $\delta_5 = 40\,\text{ms}$. This estimate is conservative and we observe stability in simulations for far fewer iterations.

4.6 Communication Requirements

The only step of dRTI that requires communication is the z-update of ADMM in Step 11 of Algorithm 4.3. If the cooperative OCP is formulated as a partially separable NLP using trajectory copies as discussed in Chapter 2, then this update can be performed as an averaging procedure which only requires neighbor-to-neighbor communication. Specifically, one can choose between different reformulations for rewriting the OCP as a partially separable NLP such that dSQP can be applied. The reformulation discussed in Example 2.1 assigns state copies w_{ij} of x_i to the out-neighbors of subsystem $i \in \mathcal{S}$, which results in an ADMM averaging procedure with two communication rounds per iteration, cf. Example 2.2. This kind of reformulation is applied to the inverted pendulum ex-

ample considered in this chapter.

An alternative would be to additionally assign copies v_{ij} of x_i to subsystem $i \in \mathcal{S}$, which leads to an ADMM averaging procedure with only one communication round per iteration, cf. Section 2.2.3. The stability guarantees of Theorem 4.2 apply to both reformulations as long as the respective assumptions on NLP (3.1) hold. From a practical point of view, the different reformulations result in different OCP dimensions with potentially different SQP convergence speeds. Consequently, the number of sufficient ADMM iterations $l_{\max}$ will likely also differ between both reformulations. It warrants further practical investigation which of the reformulations indeeed results in faster convergence.

4.7 Comparison to the Literature

There exist further approaches which adapt RTI ideas to DMPC and we briefly comment on their similarities and differences to dRTI as summarized in Figure 4.3. A distributed RTI scheme for nonlinear DMPC is discussed in the context of wind farm control in [Gros 2014]. Therein, one iteration of a second-order dual decomposition method is applied to the centralized OCP in each control step. This requires each subsystem to solve one small-scale QP, which can be done in parallel for all subsystems. A centralized coordinator then applies one Newton step to the centralized dual function and communicates the update for the coupling Lagrange multipliers to the subsystems. Conceptually, the approach in [Gros 2014] and dRTI thus both apply one Newton step to the OCP in each control step. Notably, the scheme of [Gros 2014] only requires two communication rounds per control step to perform an *exact* Newton step, whereas dRTI requires multiple ADMM iterations on the inner level. However, this comes at the cost of a centralized coordinator for performing the second-order dual update in [Gros 2014].

A further bi-level decentralized algorithm for real-time nonlinear DMPC is OTSA [Hours and Jones 2016]. Similar to dRTI, OTSA completes a fixed number of optimizer iterations per NMPC step and proves closed-loop optimizer convergence for sufficiently many inner iterations and suffi-

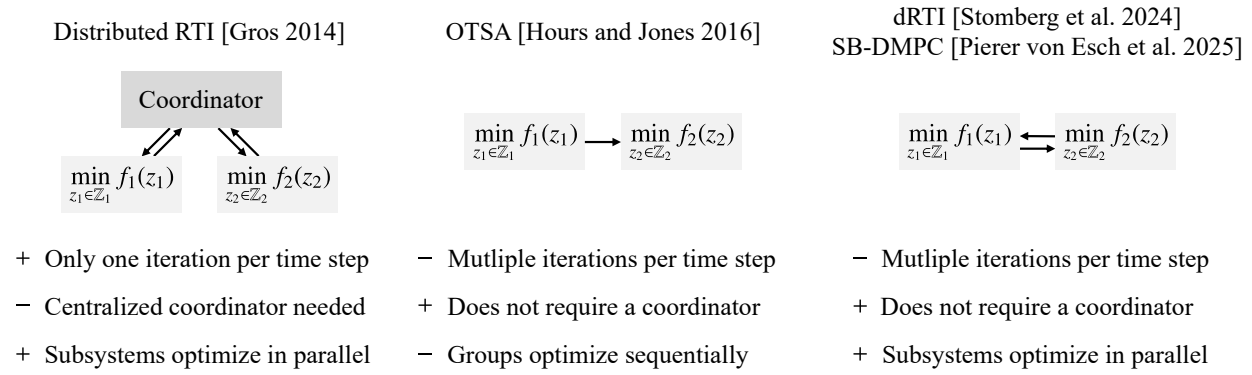


Figure 4.3: Comparison of the dRTI framework to related approaches.

cient proximity between subsequent state measurements. OTSA assumes the objective and equality constraints to be polynomial and assigns subsystems into groups which run sequentially. In contrast, dRTI covers three times continuously differentiable NLP components and allows subsystems to execute expensive computations in parallel. Moreover, Theorem 4.2 proves the stability of system *and* optimizer, whereas [Hours and Jones 2016, Theorem 5] assumes stability of the dynamical system to infer optimizer convergence. We note, however, that we here use the more recent centralized RTI stability results from [Zanelli et al. 2021] and that stability of the system-optimizer dynamics presumably may also be derived for OTSA thanks to its linear convergence.

In addition to the above algorithms for discrete-time control, a Sensitivity-Based DMPC (SB-DMPC) algorithm for the cooperative control of nonlinear systems in continuous time is presented in [Pierer von Esch et al. 2025b]. The theoretical stability analyses of SB-DMPC and dRTI were developed concurrently, with the preprint to [Pierer von Esch et al. 2025b] appearing shortly after the preprint to [Stomberg et al. 2025a]. Conceptual similarities between SB-DMPC and dRTI are that both schemes are decentralized, require multiple iterations per control step, and allow subsystems to optimize in parallel. Main differences refer to the types of problems solved on a subsystem level and the number of communication rounds per time step. Per iteration, SB-DMPC requires each subsystem to solve a nonlinear OCP and relies on one communication round. In contrast, subsystems in dRTI only solve convex QPs, but, depending on the NLP formulation, one or two communication rounds are performed in each ADMM iteration, cf. Section 4.6. From a theoretical point of view, the stability analyses of SB-DMPC and dRTI follow similar arguments as both prove linear convergence of the optimizer to infer closed-loop stability under stabilizing OCP designs. An advantage of SB-DMPC is that it considers coupled dynamics in continuous time and thus avoids difficulties related to sparse discretizations, cf. Remark 4.1. A drawback of the theoretical guarantees of SB-DMPC is that they do not consider state constraints, but require compact input constraints which are assumed to be decoupled and convex. Moreover, the dRTI guarantees presented in this section quantify the number of sufficient ADMM iterations whereas the guarantees for SB-DMPC only show the qualitative existence of a sufficiently short prediction horizon and sufficient iteration number which guarantee stability. However, the latter point likely carries little practical relevance as the quantitative iteration bound for $l_{\max}$ in dRTI is conservative. Of greater importance is the fact that both SB-DMPC and dRTI enjoy closed-loop stability guarantees for nonlinear DMPC without centralized coordination.

4.8 Summary

This chapter has presented the dRTI framework for cooperative nonlinear DMPC based on an adapted version of dSQP with a constant number of iterations [Stomberg et al. 2025a]. The resulting closed-loop system-optimizer dynamics are stable if sufficiently many ADMM iterations are performed and if the control sampling interval is sufficiently small. To the best of our knowledge,

Theorem 4.2 thus is the first stability guarantee for system *and* optimizer in cooperative nonlinear DMPC which proves, instead of assuming, decentralized optimizer convergence and which does not require a centralized coordinator or feasible initializations. Moreover, a numerical example with coupled inverted pendulums demonstrates the framework's ability to control open-loop unstable constrained nonlinear systems with coupled dynamics. However, the derived conditions for guaranteeing stability are conservative as they estimate an excessive number of ADMM iterations for the considered pendulum example. Performing a large number of optimizer iterations would, however, contradict the fundamental idea of real-time iterations and threaten real-time feasibility, especially in the presence of communication delays. Thus, the next chapter analyzes the practical performance of dRTI for robotic systems in distributed computing environments.

5 Application to Multi-Robot Formation Control

This chapter complements the theoretical analysis of dRTI presented in the previous chapter with experimental results for multi-robot formation control. We first discuss the DMPC design and give an overview of the chapter results. Section 5.2 and 5.3 then present experiments with mobile robots and hovercraft moving on an air hockey table, respectively. Finally, we discuss the results and their implication for control performance and real-time feasibility. This chapter is closely based on the formation control of mobile robots in [Stomberg et al. 2023] and holonomic hovercraft in [Stomberg et al. 2025c].

5.1 Problem Statement and Chapter Overview

We consider position-based formation control of a robotic swarm $\mathcal{S} = \{1, \dots, S\}$. The control task is formalized as a setpoint stabilization problem

$$\lim_{t \to \infty} \left\| x(t) - x^{\mathrm{d}} \right\| = 0 \quad \text{for the centralized state} \quad x \doteq (x_1, \dots, x_S),$$

where the reference state x^{d} encodes the formation by specifying a desired absolute position for each robot $i \in \mathcal{S}$. The scheme exhibits three common characteristics for position-based formation control [Oh et al. 2015]. First, all robots share a global coordinate system. Second, each robot measures its absolute position within the global coordinate system. Third, while communication between robots is in principle not necessary for solving the above setpoint stabilization problem, the communication between robots enhances performance compared to decentralized control. Here, the improved performance results from inter-robot collision avoidance constraints which are facilitated by communication between neighbors.

Collision and Obstacle Avoidance

Let $p_{\mathrm{x},i}$ and $p_{\mathrm{y},i}$ denote the position x- and y-coordinates of the geometric center of robot $i \in \mathcal{S}$. To prevent collisions between neighboring robots and with obstacles, we implement the collision avoidance constraints

$$\left\| (p_{\mathrm{x},i}, p_{\mathrm{y},i}) - (p_{\mathrm{x},j}, p_{\mathrm{y},j}) \right\|_2^2 \geq d_{\min}^2 \quad \text{for all } j \in \mathcal{N}_i^{\mathrm{in}}, \quad \text{for all } i \in \mathcal{S}, \tag{5.1a}$$

$$\left\| (p_{\mathrm{x},i}, p_{\mathrm{y},i}) - (p_{\mathrm{x,obs}}(t), p_{\mathrm{y,obs}}(t)) \right\|_2^2 \geq d_{\min}^2 \quad \text{for all } i \in \mathcal{S}. \tag{5.1b}$$

Here, $d_{\min} > 0$ is a specified minimum distance and $p_{\mathrm{x,obs}}$ and $p_{\mathrm{y,obs}}$ is the obstacle position at sampling time t. The constraints (5.1) are non-convex state constraints and, together with the linear robot dynamics to be presented in Sections 5.2 and 5.3, yield a non-convex Quadratically Constrained Quadratic Program (QCQP) as OCP. In the absence of the collision avoidance constraints, the OCP is a convex QP. Throughout this chapter, the considered robots have a circular shape such

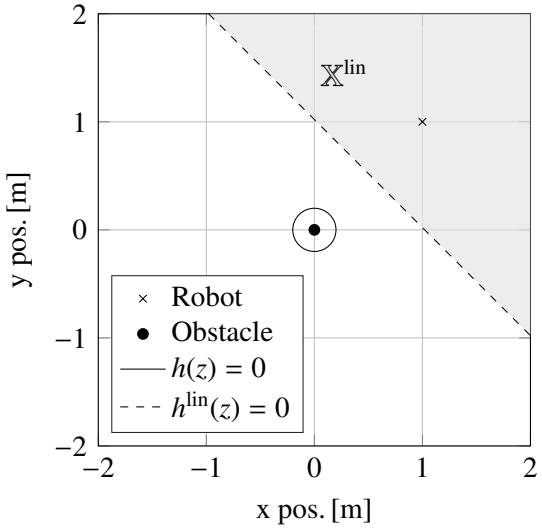


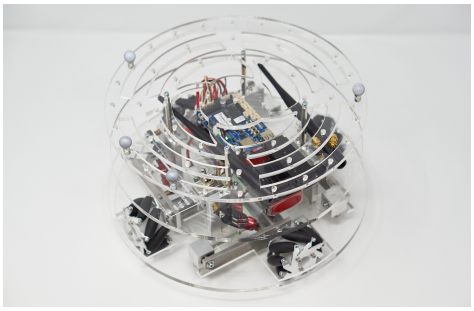
Figure 5.1: Linearization of the non-convex collision avoidance constraints with dSQP. The linearized constraint set $\mathbb{X}^{\text{lin}}$ is shaded in gray and is bounded by a separating hyperplane between the robot and the obstacle.

that the constraints (5.1) are a simple and natural choice. When solved with dSQP, the nonlinear inequality constraints $h(z)$ are linearized to obtain a state constraint set $\mathbb{X}^{\text{lin}} \doteq \{h^{\text{lin}}(z) \leq 0\}$. Figure 5.1 illustrates the linearization for a robot position $(p_x, p_y) = (1, 1)\,\text{m}$, an obstacle at $(p_{\text{x,obs}}, p_{\text{y,obs}}) = (0, 0)\,\text{m}$, and a minimum distance $d_{\text{min}} = 0.2\,\text{m}$. The boundary of the linearized constraint forms a separating hyperplane between the robot and the obstacle. Thus, while the constraint formulation (5.1) is relatively simple, the application of dSQP numerically recovers established collision avoidance frameworks such as the on-demand collision avoidance and separating hyperplane approaches in [Luis and Schoellig 2019; Van Parys and Pipeleers 2017]. The figure further highlights the conservative nature of the linearization. On the one hand, this takes away one of the main advantages of the constraints (5.1) by shrinking the feasible set of the subsystem QPs. On the other hand, however, the conservatism is more forgiving with respect to the consensus error of ADMM under early termination. Finally, notice that the separating hyperplane in Figure 5.1 is drawn for fixed robot and obstacle positions. In contrast, a closed-loop DMPC implementation produces one hyperplane for each *predicted* state over the horizon and further updates the hyperplanes in each SQP iteration.

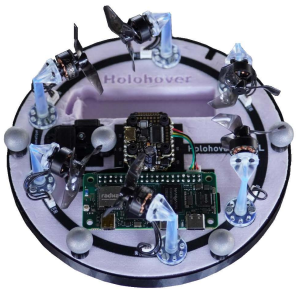
The hard collision avoidance constraints (5.1) may become infeasible due to inevitable disturbances in experiments and the absence of tailored terminal ingredients in our OCP design. Therefore, we introduce slack variables and implement the collision avoidance via soft—always feasible—constraints [Luis and Schoellig 2019] and we found the following approach to work well. We add one scalar slack variable per robot for the inter-robot constraints (5.1a),

$$\left\|(p_{\text{x},i}[\tau], p_{\text{y},i}[\tau]) - (p_{\text{x},j}[\tau], p_{\text{y},j}[\tau])\right\|_2^2 + s_i \geq d_{\text{min}}^2, \quad s_i \geq 0, \quad \text{for all} \quad j \in \mathcal{N}_i^{\text{in}},\ i \in \mathcal{S},$$

where τ is the time index in the OCP prediction. A main advantage of choosing a scalar slack



(a) Custom mobile robot at the Institute of Engineering and Computational Mechanics, University of Stuttgart, Germany. Picture by Henrik Ebel [Stomberg et al. 2023].



(b) Holonomic hovercraft "Holohover" at the Automatic Control Laboratory, EPFL, Switzerland. Picture by Roland Schwan.

Figure 5.2: Robot hardware used in the formation control experiments.

variable for each subsystem is the reduced OCP dimension compared to soft constraints with one slack variable per constraint, cf. [Maciejowski 2002, Equation 3.91]. However, we found one slack variable per constraint to work better for the obstacle avoidance constraints (5.1b),

$$\left\|(p_{\mathrm{x},i}[\tau], p_{\mathrm{y},i}[\tau]) - (p_{\mathrm{x,obs}}(t), p_{\mathrm{y,obs}}(t))\right\|_2^2 + s_i^{\mathrm{obs}}[\tau] \geq d_{\min}^2, \quad s_i^{\mathrm{obs}}[\tau] \geq 0, \quad \text{for all} \quad i \in \mathcal{S},\ \tau \in \mathbb{I}_{[1,N]}.$$

We penalize the slacks by adding the quadratic terms cs_i^2 and $\sum_{\tau=1}^{N} c\left(s_i^{\mathrm{obs}}[\tau]\right)^2$ with $c = 10^6$ to the OCP objective of subsystem $i \in \mathcal{S}$.

The rest of this chapter presents two series of experiments and their respective OCP designs, dRTI algorithms, experimental setups, and results. Section 5.2 considers custom mobile robots as the one shown in Figure 5.2a [Ebel and Eberhard 2022] and Section 5.3 describes experiments with multiple hovercraft as the one depicted in Figure 5.2b [Schwan et al. 2024]. The fundamental OCP and dRTI designs are similar for all experiments and Figure 5.3 highlights key features, similarities, and differences of Sections 5.2 and 5.3.

5.2 Experiments with Mobile Robots

This section summarizes formation control experiments with four mobile robots as the one depicted in Figure 5.2a. The section is based on the journal article [Stomberg et al. 2023].

5.2.1 OCP Design and dRTI Algorithm

The mobile robots in Figure 5.2a are equipped with Proportional-Integral-Derivative (PID) controllers which command the actuators to track a defined velocity reference. The prediction model used in the DMPC design thus is given by the single integrator dynamics

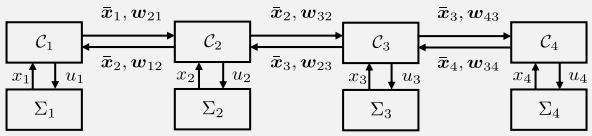
$$x_i(t+1) = x_i(t) + \delta \cdot u_i(t) \quad \text{for all} \quad i \in \mathcal{S},$$

Chapter 5: Robot Formation Control

I. Experiments with four robots/hovercraft

II. Distributed computation: neighboring controllers run on distinct machines and exhibit communication delay

III. Compensation of computational delay in the OCP



Section 5.1: Mobile Robots

1. Offboard execution on workstation PCs
2. Communication via Ethernet
3. Comparison: with vs. without collision avoidance
4. Algorithm 5.1: synchronous ADMM
5. ADMM averaging step: $z^{k,l+1} = M_{\text{avg}} y^{k,l+1}$

Section 5.2: Hovercraft on Air Hockey Table

1. Comparison: offboard vs. onboard execution
2. Comparison: Ethernet (offboard) vs. WiFi (onboard)
3. Comparison: with vs. without dynamic obstacle
4. Algorithm 5.2: synchronous ADMM with fallback to asynchronous ADMM in case of package loss/delay
5. ADMM averaging step: $z^{k,l+1} = M_{\text{avg}}(y^{k,l+1} + \underbrace{\gamma^{k,l}/\rho}_{\text{for asynchronous execution}})$

Figure 5.3: Overview of Chapter 5. The block diagram is taken from [Stomberg et al. 2023] and shows the path graph coupling structure underlying all experiments with four cooperatively controlled robots/hovercraft Σ_i. The controllers C_i exchange predicted state trajectories $\bar{\boldsymbol{x}}_i$ and copies $\boldsymbol{w}_{ij}$ with neighbors.

where $x_i = (p_{\text{x},i}, p_{\text{y},i})$ is the position and $u_i \in \mathbb{R}^2$ is the velocity of robot i. Let $s \doteq (s_1, \ldots, s_S)$ denote the centralized slack variable. The OCP with soft inter-robot collision avoidance constraints reads

$$\min_{\boldsymbol{x},\boldsymbol{u},s} \sum_{i\in\mathcal{S}} \left(\sum_{\tau=0}^{N-1} \ell_i\left(x[\tau], u_i[\tau]\right) + V_{\text{f},i}(x[N]) + c s_i^2 \right) \tag{5.2a}$$

subject to for all $i \in \mathcal{S}$

$$x_i[\tau+1] = x_i[\tau] + \delta \cdot u_i[\tau] \qquad \forall \tau \in \mathbb{I}_{[0,N-1]}, \tag{5.2b}$$

$$(x_i[0], u_i[0]) = (x_i(t), u_i(t)), \tag{5.2c}$$

$$-\hat{u}_i \le u_i[\tau] \le \hat{u}_i \qquad \forall \tau \in \mathbb{I}_{[0,N-1]}, \tag{5.2d}$$

$$d_{\min}^2 \le \left\| x_i[\tau] - x_j[\tau] \right\|_2^2 + s_i \quad \forall j \in \{i-1\} \cap \mathcal{S}, \quad \forall \tau \in \mathbb{I}_{[0,N]}, \tag{5.2e}$$

$$0 \le s_i. \tag{5.2f}$$

The stage cost and terminal penalty for robot $i \in \mathcal{S}$ are given by

$$\ell_i(x, u_i) \doteq \sum_{j\in\mathcal{N}_i^{\text{in}}} \frac{1}{2}(x_i - x_i^{\text{d}}) Q_{ij} (x_j - x_j^{\text{d}}) + \frac{1}{2} u_i^\top R_i u_i \quad \text{and} \quad V_{\text{f}_i} \doteq \sum_{j\in\mathcal{N}_i^{\text{in}}} \frac{1}{2}(x_i - x_i^{\text{d}}) P_{ij} (x_j - x_j^{\text{d}}). \tag{5.3}$$

The cost matrices are chosen as $Q_{11} = Q_{22} = Q_{33} = \text{diag}(20, 20)$, $Q_{44} = \text{diag}(10, 10)$, $Q_{ij} = \text{diag}(-10, -10)$ for all $j \in \{i-1, i+1\} \cap \mathcal{S}$, $R_i = \text{diag}(1, 1)$, and $P_{ij} = Q_{ij}$ for all $i, j \in \mathcal{S}$. The

set of in-neighbors of robot i thus is $\mathcal{N}_i^{\text{in}} \doteq \{i-1, i+1\} \cap \mathcal{S}$. Each robot is subject to input box constraints with the maximum velocity $\hat{u}_i \doteq (0.2, 0.2)$ m/s. The horizon is chosen as $N = 7$ and the control sampling interval is $\delta = 200$ ms.

Remark 5.1 (Compensation of computational delay)**.** *Constraint* (5.2c) *specifies an initial condition on the state and the input of robot i, which allows to compensate the computational delay of solving the OCP. Decentralized algorithms such as ADMM and dSQP may take up a substantial part of the control sampling interval, especially in the presence of communication delays. Thus, we solve OCP* (5.2) *for the input* $u[1]$ *to be applied in the next control step [Findeisen 2006]. Therefore, dRTI is real-time feasible as long as the total OCP solve time is below the sampling interval* δ*.* □

To apply dRTI, we translate OCP (5.2) into a consensus problem in partially separable form like (3.1) via state copies as illustrated in Example 2.1. Recall that this assigns the predicted copy trajectories $\boldsymbol{w}_i \doteq \left(\boldsymbol{w}_{ji}\right)_{j \in \mathcal{N}_i^{\text{out}}}$ as decision variables to subsystem i, where $\boldsymbol{w}_{ji}$ is a copy of $\boldsymbol{x}_j$. The OCP additionally contains the slack variable s_i for all $i \in \mathcal{S}$ and the two decision variable blocks read

$$y_i \doteq (\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{w}_i, s_i) \in \mathbb{R}^{n_i} \quad \text{and} \quad z_i \doteq (\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{u}}_i, \bar{\boldsymbol{w}}_i, \bar{s}_i) \in \mathbb{R}^{n_i} \quad \text{for all} \quad i \in \mathcal{S}.$$

Algorithm 5.1 summarizes the dRTI steps, where ADMM is decentralized via the averaging Step 12.

5.2.2 Experimental Setup

Figure 5.4 sketches the hardware setup used in the experiments at the Institute of Engineering and Computational Mechanics at the University of Stuttgart. Each of the four robots is assigned one workstation PC for running Algorithm 5.1. The PCs run Debian Linux and are equipped with either an Intel Core i7-9750H (2.6 GHz), Intel Xeon E5530 (2.4 GHz), or Intel Xeon E31245 (3.3 GHz) processor. Ethernet is used to communicate the predicted state trajectories in Steps 11 and 13 between PCs and to receive the position measurements in Step 3. Communicated variables are insinuated in Figure 5.4 and the coupling graph in Figure 5.3 details the exchanged variables between subsystems. Recall that the states x_i here are the robot positions. Consequently, the motion capturing system provides a full state measurement and no additional observer is required. The robot poses are measured by an Optitrack system using six Prime 13 W cameras and the commercial Motive software. Custom software publishes the state measurements via User Datagram Protocol (UDP) multicast and the Lighweight Communications and Marshalling (LCM) library for delivery to the DMPC PCs [Huang et al. 2010]. The DMPC control input is sent to each robot wirelessly using UDP multicast and the LCM library. Further details of our open-source C++ implementation are summarized in Appendix B.1.[1]

The robots have a cylindrical shape with a diameter of 290 mm and a detailed description of the design is given in [Ebel 2021]. As mentioned above, we control the team of robots via a bi-level

[1] `https://github.com/OptCon/dmpc_rto`

Algorithm 5.1 Synchronous dRTI for cooperative DMPC on robot i [Stomberg et al. 2023]

Initialization: $t = 0, u_i(0), x_i^{\mathrm{d}}, z_i^0(0), \gamma_i^0(0) = E_i^\top \lambda^0, k_{\max}, l_{\max}, \rho$
for NMPC step $t = 0, 1, \ldots,$ **do**
  estimate the current state $x_i(t)$ and apply the control input $u_i(t)$
  update the OCP initial condition (5.2c) with $x_i(t)$ and $u_i(t)$
  initialize dSQP by setting z_i^0 and γ_i^0 to $z_i^{k_{\max}}(t-1)$ and $\gamma_i^{k_{\max}}(t-1)$ shifted by the time index
  for SQP iteration $k = 0, \ldots, k_{\max} - 1$ **do**
    evaluate ∇f_i^k, g_i^k, ∇g_i^k, h_i^k, and ∇h_i^k
    initialize ADMM by setting $z_i^{k,0} = z_i^k$ and $\gamma_i^{k,0} = \gamma_i^k$
    for ADMM iteration $l = 0, \ldots, l_{\max} - 1$ **do**
      solve subsystem QP $y_i^{k,l+1} = \underset{y_i \in \mathbb{Z}_i^k}{\operatorname{argmin}} L_{\rho,i}^k(y_i, z_i^{k,l}, \gamma_i^{k,l})$
      send $\boldsymbol{w}_{ji}^{k,l+1}$ to all $j \in \mathcal{N}_i^{\mathrm{in}}$ and receive $\boldsymbol{w}_{ij}^{k,l+1}$ from all $j \in \mathcal{N}_i^{\mathrm{out}}$
      compute average state trajectory $\bar{\boldsymbol{x}}_i^{k,l+1} = \dfrac{1}{|\mathcal{N}_i^{\mathrm{out}}| + 1} \left(\boldsymbol{x}_i^{k,l+1} + \sum_{j \in \mathcal{N}_i^{\mathrm{out}}} \boldsymbol{w}_{ij}^{k,l+1} \right)$
      send average $\bar{\boldsymbol{x}}_i^{k,l+1}$ to all $j \in \mathcal{N}_i^{\mathrm{out}}$ and receive average $\bar{\boldsymbol{x}}_j^{k,l+1}$ from all $j \in \mathcal{N}_i^{\mathrm{in}}$
      set $\bar{\boldsymbol{u}}_i^{k,l+1} = \boldsymbol{u}_i^{k,l+1}$, $\bar{\boldsymbol{w}}_{ji}^{k,l+1} = \bar{\boldsymbol{x}}_j^{k,l+1}$ for all $j \in \mathcal{N}_i^{\mathrm{in}}$, and $\bar{s}_i^{k,l+1} = s_i^{k,l+1}$
      form $z_i^{k,l+1} = (\bar{\boldsymbol{x}}_i^{k,l+1}, \bar{\boldsymbol{u}}_i^{k,l+1}, \bar{\boldsymbol{w}}_i^{k,l+1}, \bar{s}_i^{k,l+1})$
      $\gamma_i^{k,l+1} = \gamma_i^{k,l} + \rho(y_i^{k,l+1} - z_i^{k,l+1})$
    end for
    $z_i^{k+1} = z_i^{k,l_{\max}}$, $\gamma_i^{k+1} = \gamma_i^{k,l_{\max}}$
  end for
  extract $u_i[1]$ from $z_i^{k_{\max}}(t)$
  set $u_i(t+1) = u_i[1]$ to compensate delay
end for

architecture with the cooperative DMPC scheme on the upper level and PID controllers for each robot on the lower level. The DMPC scheme computes a desired velocity $u_i \in \mathbb{R}^2$ at a sampling frequency of 5 Hz. The PID controllers run on microcontrollers onboard the robots and command the motors at a sampling frequency of 100 Hz to track the desired velocity. Even though the four robots share the same design, manufacturing imperfections result in different dynamics for each robot [Eschmann et al. 2021]. Moreover, the true robot dynamics are more complicated than the single integrator model, because of friction, wheel-floor contacts, and the PID controllers on the inner level.

To summarize, the hardware setup exhibits three important characteristics which would also affect real-world DMPC implementations. First, the communication between PCs incurs communication delays and losses. Second, the imperfect robot hardware results in non-negligible model-plant mismatch. Third, the heterogeneous computing power of the PCs causes the DMPC programs to wait for neighbors to stay synchronized, adding to the overall DMPC execution time.

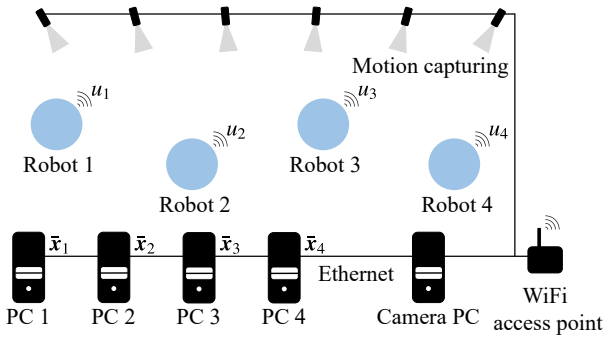


Figure 5.4: Hardware setup with four mobile robots. Each robot is assigned one workstation PC to run the dRTI Algorithm 5.1. Optimizer iterates are exchanged via Ethernet and control signals are sent via WiFi. Figure adapted from [Stomberg et al. 2023].

5.2.3 Experimental Results: Linear-Quadratic DMPC and ADMM

We first test linear-quadratic DMPC without the collision avoidance constraints (5.2e). The OCP is a convex QP without slack variables and, in centralized form, has $n = 216$ decision variables, $n_g = 72$ subsystem equality constraints, $n_h = 56$ inequality constraints, and $n_c = 96$ coupling constraints. Since the OCP is a QP, dSQP reduces to ADMM so that the SQP Steps 6–8 and 18–19 in Algorithm 5.1 as well as the updates on the slack variables s are omitted. We tune the ADMM penalty parameter by choosing $\rho = 1$ from the set $\{0.1, 1, 10, 100\}$ for fast convergence to the OCP minimizer and we choose $l_{\max} = 5$ iterations per MPC step. Furthermore, we initialize the dRTI algorithm with $u_i(0) = 0\,\mathrm{m/s}$ and $z_i^0(0) = \gamma_i^0(0) = 0$ for all $i \in \mathcal{S}$ at $t = 0$.

Figure 5.5 illustrates the scenario and a video recording is available online in the supplementary material of the article [Stomberg et al. 2023].[2] The first robot drives along a rectangular path encoded via the desired position $x_1^{\mathrm{d}}(t)$. The remaining robots follow their predecessor at a distance $\Delta x^{\mathrm{d}} \doteq (-0.4, 0)\,\mathrm{m}$, i.e., $x_i^{\mathrm{d}}(t) \doteq x_{i-1}^{\mathrm{d}}(t) + \Delta x^{\mathrm{d}}$ for all $i \in \{2, 3, 4\}$. The closed-loop trajectories and optimizer error for the control input are shown in Figure 5.6, where the left and right columns refer to the x- and y-coordinates in the global reference frame, respectively. The DMPC controllers track the desired setpoints, meet the input box constraints, and achieve a cm/s accuracy in the commanded velocity compared to the optimal control $u^\star$ found by ipopt. This illustrates the system-optimizer convergence discussed in Chapter 4 and aligns with previous findings which showed that few ADMM iterations per control step can suffice in closed-loop control [Burk et al. 2021b; Stomberg et al. 2022a; Van Parys and Pipeleers 2017].

Table 5.1 summarizes the median and maximum ADMM execution times recorded in two experiments with a total duration of approximately three minutes. The top row reports the overall time ADMM took to solve the OCP in each MPC step using $l_{\max} = 5$ iterations. The maximum solve time of 22.27 ms is below the control sampling interval $\delta = 200\,\mathrm{ms}$ and is thus compensated by the MPC scheme. The remaining rows report the times for one ADMM iteration, divided into

[2] https://doi.org/10.1016/j.conengprac.2023.105579

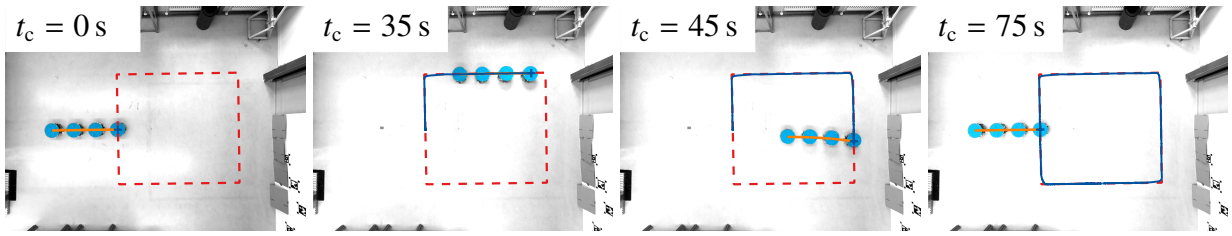


Figure 5.5: Rectangle scenario using linear-quadratic DMPC based on ADMM: camera images and rectangular path (dashed line), closed-loop trajectory of robot one (solid blue line), and current formation (solid orange line). Pictures by Henrik Ebel [Stomberg et al. 2023].

Table 5.1: ADMM solve time divided into the main steps of Algorithm 5.1 [Stomberg et al. 2023].

	Median [ms]	Maximum [ms]
Solve OCP (5.2) with ADMM	6.61	22.27
Time per ADMM iteration	0.98	16.71
Solve one subsystem QP (Step 10)	0.09	14.57
Communicate $\boldsymbol{w}$ (Steps 11–12)	0.29	11.92
Communicate $\bar{\boldsymbol{x}}$ (Step 13)	0.25	11.63

the most expensive steps. For the communication of y in Step 11 and z in Step 13, the times span the following five substeps: (i) write the optimizer variables into an LCM message, (ii) publish the LCM message, (iii) wait for messages from neighbors, and (iv) cache the received optimizer iterates. The execution time of the y-communication further includes the evaluation of the averaging Step 12. The table shows that the communication delay here dominates the ADMM exeuction time. This can also be seen from Table 5.2 which reports the relative execution times each robot spends on either solving the subsystem QP or waiting/communicating. The differences between the individual robots highlight the varying computational power of the employed workstation PCs.

5.2.4 Experimental Results: Collision Avoidance and dSQP

We next consider nonlinear DMPC including the inter-robot collision avoidance constraints (5.2) with minimum distance $d_{\min} = 0.4\,\text{m}$. The remaining OCP parameters are left unchanged and

Table 5.2: Fraction of the ADMM execution time spent computing and communicating [Stomberg et al. 2023].

Robot	1	2	3	4
Solve subsystem QP (Step 10)	37.37 %	33.38 %	14.52 %	15.51 %
Communicate (Steps 11–13)	62.10 %	66.24 %	85.23 %	84.15 %

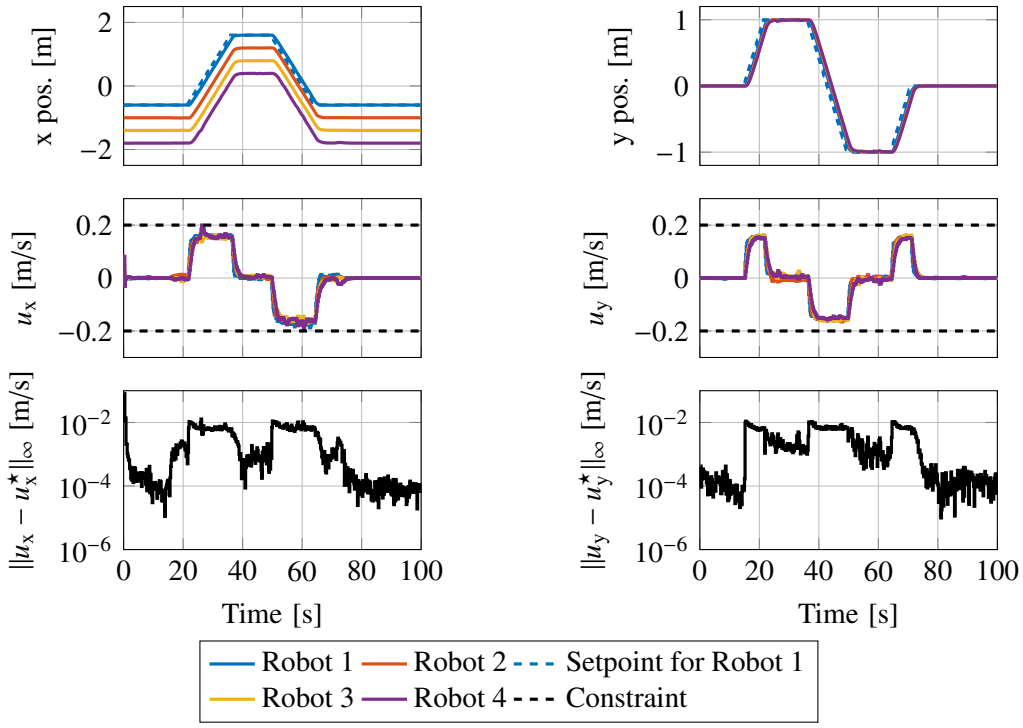


Figure 5.6: Rectangle scenario using linear-quadratic DMPC based on ADMM: closed-loop trajectories and optimizer error. Figure adapted from [Stomberg et al. 2023].

the resulting non-convex QCQP is solved using dSQP with $k_{\max} = 5$ and $l_{\max} = 3$ iterations. We reformulate the OCP as a partially separable QCQP as demonstrated in Example 2.1 and the centralized QCQP has $n = 219$ decision variables, $n_g = 72$ equality constraints, $n_h = 83$ inequality constraints, and $n_c = 96$ coupling constraints. We again tune the penalty parameter for fastest convergence to an OCP minimizer found by ipopt and choose $\rho = 1$ from the set $\rho = \{0.1, 1, 10, 100\}$. To avoid costly derivative computations and regularization procedures, we use the GN Hessian approximation $H_i = \nabla^2_{z_i z_i} f_i$ for all $i \in \mathcal{S}$.

Figure 5.7 shows a formation-change scenario where the center robots swap positions when transitioning between the formations marked by the green and red crosses. As for the experiments on linear-quadratic DMPC, a video recording is available online [Stomberg et al. 2023]. The considered scenario is more challenging than the above rectangle maneuver. Indeed, the above linear-quadratic DMPC scheme without collision avoidance constraints fails to perform the formation change in simulations because robots two and three collide. The corresponding closed-loop trajectories are depicted in Figure 5.8 and show that the input constraints are active after each setpoint change. Moreover, the bottom plots show how dSQP converges to the minimizer found by ipopt. However, the optimizer accuracy is reduced at the beginning of the experiment and after the setpoint change at $t_c = 20\,\text{s}$, because the dSQP initialization is worse in these NMPC steps.

Table 5.3 presents the median and maximum times recorded in twelve experiments with a total duration of twelve minutes. The top row refers to the total time needed by dSQP to solve the

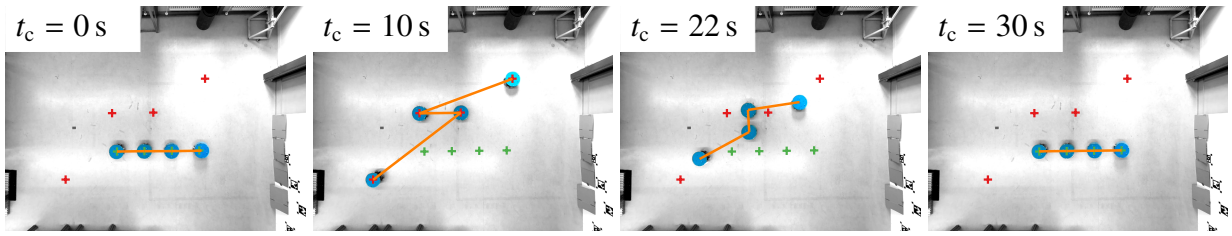


Figure 5.7: Formation-change scenario using nonlinear DMPC based on dSQP: camera images and formation one (green crosses), formation two (red crosses), and current formation (solid orange line). Pictures by Henrik Ebel [Stomberg et al. 2023].

Table 5.3: dSQP solve time divided into the main steps of Algorithm 5.1 [Stomberg et al. 2023].

	Median [ms]	Maximum [ms]
Solve OCP (5.2) with dSQP	33.81	53.06
Time per dSQP iteration	6.23	24.57
Build subsystem QP (Step 7)	0.30	1.09
Run ADMM per dSQP iter	5.99	24.29
Time per ADMM iteration	2.01	15.37
Solve subsystem QP (Step 10)	0.81	12.40
Communicate $\boldsymbol{w}$ (Step 11–12)	0.64	12.17
Communicate $\bar{\boldsymbol{x}}$ (Step 13)	0.32	11.53

OCP and the remaining rows refer to the most expensive steps per optimizer iteration. The maximum solve time per NMPC step is less than 54 ms, which highlights the real-time capabilities of the algorithm. Most of the execution time is spent on ADMM and the derivative computation is comparatively fast thanks to the GN Hessian approximation. Table 5.4 further shows the relative solve time required by each subsystem for building and solving the subsystem QP and for communicating. While robots one, three, and four spend more time on communication than on local computations, this is not the case for robot two. On the one hand, this is due to the weaker computing power of PCs one and two compared to PCs three and four. On the other hand, recall from the OCP formulation (5.1) that, in this experiment, only robots two to four are subject to collision avoidance constraints, whereas robot one has no state constraints. As a result, the subsystem QP for robot one is less complex than for the other robots and, because of the heteregenous computing power, robot two thus requires longer solve times than the other robots.

5.3 Embedded DMPC and Experiments with Hovercraft

We continue with formations consisting of four hovercraft like the one shown in Figure 5.2b. The hovercraft move on an air hockey table and this section is based on the conference paper [Stomberg et al. 2025c].

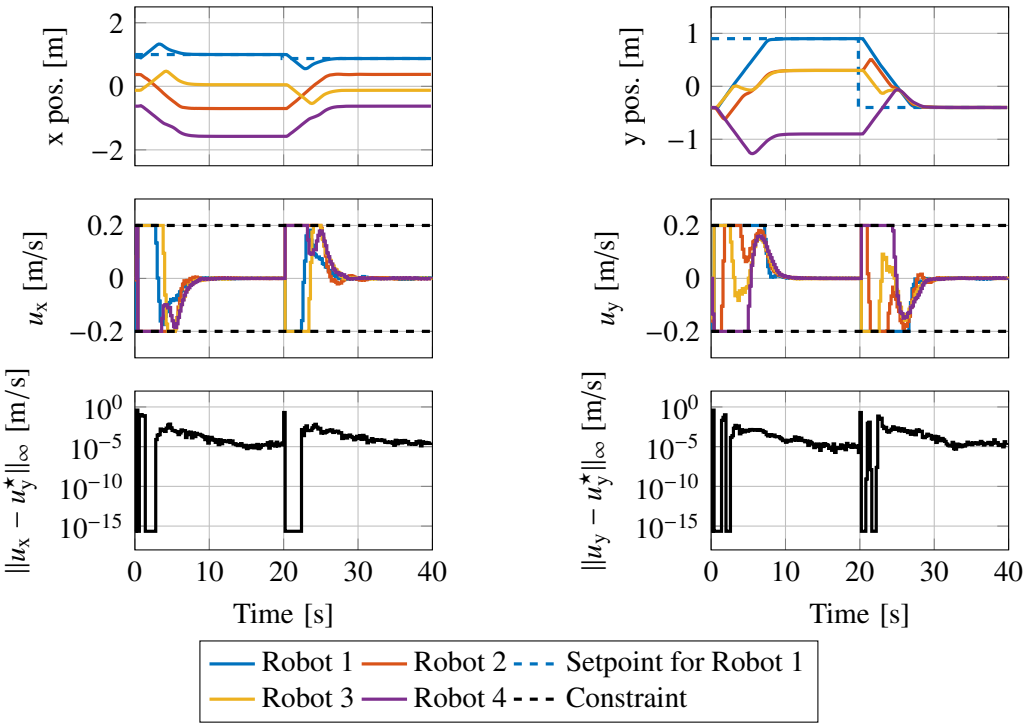


Figure 5.8: Formation-change scenario using nonlinear DMPC based on dSQP: closed-loop trajectories and optimizer error. Figure adapted from [Stomberg et al. 2023].

Table 5.4: Fraction of the dSQP solve time spent computing and communicating [Stomberg et al. 2023].

Robot	1	2	3	4
Build subsystem QP (Step 7)	5.37 %	4.72 %	3.10 %	4.34 %
Solve subsystem QP (Step 10)	34.02 %	68.49 %	41.56 %	29.46 %
Communicate (Steps 11–13)	60.23 %	26.46 %	55.11 %	65.89 %

5.3.1 OCP Design and dRTI Algorithm

Each hovercraft $i \in \mathcal{S}$ is modeled as a double integrator with state $x_i \doteq (p_{\mathrm{x},i}, p_{\mathrm{y},i}, v_{\mathrm{x},i}, v_{\mathrm{y},i}, \varphi_i, \omega_i) \in \mathbb{R}^6$ and input $u_i \doteq (u_{\mathrm{x},i}, u_{\mathrm{y},i}, u_{\varphi,i}) \in \mathbb{R}^3$, where $p_i \doteq (p_{\mathrm{x},i}, p_{\mathrm{y},i})$ denotes the position of the hovercraft center in a fixed global reference frame, φ_i is the yaw angle, and $(v_{\mathrm{x},i}, v_{\mathrm{y},i})$ and ω_i are the translational and rotational velocity, respectively. For all $i \in \mathcal{S}$, the continuous-time dynamics read

$$\dot{p}_i(t_\mathrm{c}) = v_i(t_\mathrm{c}), \quad \dot{\varphi}_i(t_\mathrm{c}) = \omega_i(t_\mathrm{c}), \quad (\dot{v}_i(t_\mathrm{c}), \dot{\omega}_i(t_\mathrm{c})) = u_i(t_\mathrm{c}), \quad x_i(0) = x_{i,0}.$$

Given a control sampling interval δ and a piecewise-constant input signal, the corresponding discrete-time dynamics in abstract form read

$$x_i(t+1) = A_i x_i(t) + B_i u_i(t), \quad x_i(0) = x_{i,0}.$$

Here, we drop the dependence of the matrices $A_i \in \mathbb{R}^{6\times6}$ and $B_i \in \mathbb{R}^{6\times3}$ on δ for notational simplicity. Every hovercraft is actuated by six propellers, each of which is driven by a brushless direct

current motor. For all $i \in \mathcal{S}$, the desired acceleration u_i is determined by the DMPC scheme and is converted into motor signals using a control allocation based on a nonlinear thrust model [Schwan et al. 2024]. The true hovercraft dynamics are more complex than the adopted double integrator model because of the actuator dynamics, inaccuracies in the thrust model, and complex aerodynamic effects. Moreover, unavoidable surface irregularities of the air hockey table add to the model-plant mismatch. To achieve offset-free tracking despite these disturbances, we adopt a constant-disturbance prediction model

$$x_i[\tau+1] = A_i x_i[\tau] + B_i u_i[\tau] + B_i d_i(t), \quad x_i[0] = x_i(t), \quad \text{for all} \quad \tau \in \mathbb{I}_{[0,N-1]}$$

with disturbance $d_i \in \mathbb{R}^3$. For all $i \in \mathcal{S}$, the augmented state $\xi_i \doteq (x_i, d_i)$ is estimated from discrete-time measurements for the pose $(p_{\mathrm{x},i}, p_{\mathrm{y},i}, \varphi_i)$ using a continuous-time Extended Kalman Filter (EKF) [Simon 2006, p. 405]. The soft-constrained OCP to be solved in step t reads

$$\min_{\boldsymbol{x},\boldsymbol{u},s,\boldsymbol{s}^{\mathrm{obs}}} \sum_{i\in\mathcal{S}} \left(\sum_{\tau=0}^{N-1} \ell_i\left(x[\tau], u_i[\tau]\right) + V_{\mathrm{f},i}(x[N]) + c s_i^2 + \sum_{\tau=0}^{N} c \left(s_i^{\mathrm{obs}}[\tau]\right)^2 \right) \tag{5.4a}$$

subject to for all $i \in \mathcal{S}$

$$x_i[\tau+1] = A_i x_i[\tau] + B_i u_i[\tau] + B_i d_i(t) \qquad \forall \tau \in \mathbb{I}_{[0,N-1]}, \tag{5.4b}$$

$$(x_i[0], u_i[0]) = (x_i(t), u_i(t)), \tag{5.4c}$$

$$-\hat{p}_i - s_i \le p_i[\tau] \le \hat{p}_i + s_i \qquad \forall \tau \in \mathbb{I}_{[0,N]}, \tag{5.4d}$$

$$-\hat{u}_i \le u_i[\tau] \le \hat{u}_i \qquad \forall \tau \in \mathbb{I}_{[0,N-1]}, \tag{5.4e}$$

$$d_{\min}^2 \le \|p_i[\tau] - p_j[\tau]\|_2^2 + s_i \quad \forall j \in \{i-1, i+1\} \cap \mathcal{S}, \quad \forall \tau \in \mathbb{I}_{[0,N]}, \tag{5.4f}$$

$$d_{\min}^2 \le \|p_i[\tau] - p_{\mathrm{obs}}(t)\|_2^2 + s_i^{\mathrm{obs}}[\tau] \qquad \forall \tau \in \mathbb{I}_{[0,N]}, \tag{5.4g}$$

$$0 \le s_i, \quad 0 \le s_i^{\mathrm{obs}}[\tau] \qquad \forall \tau \in \mathbb{I}_{[0,N]}. \tag{5.4h}$$

Here, $\boldsymbol{s}^{\mathrm{obs}} \doteq (s_1^{\mathrm{obs}}[0], s_1^{\mathrm{obs}}[1], \ldots, s_S^{\mathrm{obs}}[N-1], s_S^{\mathrm{obs}}[N])$ collects all slack variables for the obstacle avoidance constraints (5.4g) and $c = 10^6$. We test two sampling intervals $\delta = 50\,\mathrm{ms}$ and $\delta = 150\,\mathrm{ms}$ and the horizon N is chosen such that the continuous-time horizon $T \doteq N \cdot \delta \approx 1\,\mathrm{s}$, i.e., we set $N = 20$ for $\delta = 50\,\mathrm{ms}$ and $N = 7$ for $\delta = 150\,\mathrm{ms}$. For $N = 7$, the partially separable QCQP consists of $n = 568$ decision variables, $n_g = 204$ subsystem equality constraints, $n_h = 302$ inequality constraints, and $n_c = 288$ coupling constraints. For $N = 20$, we have $n = 1504$, $n_g = 516$, $n_h = 900$, and $n_c = 756$. The costs have the same structure (5.3) as in the mobile robots experiments. We set $Q_{11} = Q_{22} = Q_{33} = \mathrm{diag}(28, 28, 18, 18, 40, 18)$ and $Q_{44} = \mathrm{diag}(14, 14, 9, 9, 20, 9)$ as well as $Q_{ij} = -\mathrm{diag}(14, 14, 9, 9, 20, 9)$ for all $j \in \mathcal{N}_i^{\mathrm{in}} \doteq \{i-1, i+1\} \cap \mathcal{S}$ and $R_i = \mathrm{diag}(0.1, 0.1, 0.1)$ for all $i \in \mathcal{S}$.

In contrast to the mobile robots, the simple choice $P = Q$ is ineffective for the hovercraft and caused poor control performance in experiments with large oscillations near the setpoint. Thus, we here design dedicated terminal penalties by solving, for all $j \in \mathcal{N}_i^{\mathrm{in}}$ and for all $i \in \mathcal{S}$, the discrete-time algebraic Riccati equation (4.35) for the pair (Q_{ij}, R_i) to obtain the weight matrix

$P_{ij} \in \mathbb{R}^{6\times 6}$. Specifically, we tuned the weight matrices Q and R iteratively in simulations and experiments to dampen the eigenvalues of the centralized closed-loop system matrix $A_K \doteq A + BK$, where $A \doteq \operatorname{diag}(A_1, \dots, A_S)$, $B \doteq \operatorname{diag}(B_1, \dots, B_S)$, and where K is obtained as a byproduct when solving the discrete-time algebraic Riccati equation for the centralized system (A, B) with weight matrices $Q = [Q_{ij}]_{i,j\in\mathcal{S}}$ and $R = \operatorname{diag}(R_1, \dots, R_S)$.

The position box constraints (5.4d) with $\hat{p} \doteq (0.96, 0.42)\,\mathrm{m}$ impose a 30 mm margin to the table boundaries and the commanded accelerations are limited to $\hat{u}_i \doteq (5\,\mathrm{m/s^2}, 5\,\mathrm{m/s^2}, 15\,\mathrm{rad/s^2})$. The obstacle is a fifth hovercraft which does not participate in the cooperative DMPC scheme. The cooperating hovercraft know the current obstacle position $p_{\mathrm{obs}}(t) \doteq (p_{\mathrm{x,obs}}(t), p_{\mathrm{y,obs}}(t))$, but not the future obstacle motion. The inter-robot and obstacle collision avoidance constraints use a minimum distance $d_{\min} = 200\,\mathrm{mm}$ which, given the hovercraft diameter of 150 mm, leaves 50 mm as clearance in case of constraint violations.

The implemented dRTI algorithm is similar to the one used in the mobile robots experiments, but is summarized again in Algorithm 5.2 for a self-contained presentation. The major difference between Algorithms 5.1 and 5.2 lies in the ADMM averaging procedure in Steps 11–12. Since both algorithms implement the dRTI scheme of Chapter 4, they are intended to run synchronously. That is, subsystems are expected to wait for messages from neighbors before continuing with the computations. The C++ implementation used in the mobile robots experiments enforces a synchronous execution as it waits indefinitely for messages in Steps 11 and 13. However, this proved not viable when running dRTI onboard the hovercraft due to the longer latencies in wireless communication. Therefore, each hovercraft only spends a maximum waiting time in Steps 11 and 13 of Algorithm 5.2 and continues without having received all messages, if necessary.

Recall from Chapter 2 that the average in Step 12 of Algorithm 5.1 only solves the coordination QP for the z update in ADMM if $M_{\mathrm{avg}}\gamma = 0$, i.e., if the average of γ vanishes. For the here-considered consensus problems in partially separable form, the only decision variables that are coupled between subsystems—and must thus be averaged—are the predicted state trajectories $\bar{\boldsymbol{x}}_i$ for all $i \in \mathcal{S}$. Therefore, the condition $M_{\mathrm{avg}}\gamma^{k,l} = 0$ can be written as

$$\frac{1}{|\mathcal{N}_i^{\mathrm{out}}| + 1}\left(\gamma_{i,\boldsymbol{x}_i}^{k,l} + \sum_{j\in\mathcal{N}_i^{\mathrm{out}}} \gamma_{j,\boldsymbol{w}_{ij}}^{k,l}\right) = 0 \quad \text{for all} \quad i \in \mathcal{S}, \tag{5.5}$$

where $\gamma_{i,\boldsymbol{x}_i} \in \mathbb{R}^{(N+1)\cdot n_{x,i}}$ and $\gamma_{j,\boldsymbol{w}_{ij}} \in \mathbb{R}^{(N+1)\cdot n_{x,i}}$ are the components of γ_i and γ_j that correspond to the constraints $\boldsymbol{x}_i - \bar{\boldsymbol{x}}_i = 0$ and $\boldsymbol{w}_{ij} - \bar{\boldsymbol{w}}_{ij} = 0$, respectively. That is, the average of all multipliers in γ which correspond to the predicted state trajectory $\boldsymbol{x}_i$ or its copies $\boldsymbol{w}_{ij}$ must be zero.

Moreover, recall that the ADMM updates ensure that condition (5.5) holds as long as the subsystems are synchronized, cf. Lemma 2.1. In other words, the averaging steps in Algorithms 5.1 and 5.2 are equivalent if dRTI runs synchronously. However, once the subsystems run asynchronously, then condition (5.5) no longer holds.

Remark 5.2 (Differences between Algorithms 5.1 and 5.2.)**.** *In contrast to Algorithm 5.1, the*

Algorithm 5.2 Asynchronous dRTI for cooperative DMPC on hovercraft i [Stomberg et al. 2025c].

1: Initialization: $t = 0, u_i(0), x_i^{\mathrm{d}}, z_i^0(0), \gamma_i^0(0), k_{\max}, l_{\max}, \rho$
2: **for** NMPC step $t = 0, 1, \ldots,$ **do**
3: estimate the current state $x_i(t)$ and apply the control input $u_i(t)$
4: update the OCP initial condition (5.4c) with $x_i(t)$ and $u_i(t)$
5: initialize dSQP by setting z_i^0 and γ_i^0 to $z_i^{k_{\max}}(t-1)$ and $\gamma_i^{k_{\max}}(t-1)$ without time shift
6: **for** SQP iteration $k = 0, \ldots, k_{\max} - 1$ **do**
7: evaluate ∇f_i^k, g_i^k, ∇g_i^k, h_i^k, and ∇h_i^k
8: initialize ADMM by setting $z_i^{k,0} = z_i^k$ and $\gamma_i^{k,0} = \gamma_i^k$
9: **for** ADMM iteration $l = 0, \ldots, l_{\max} - 1$ **do**
10: solve subsystem QP $y_i^{k,l+1} = \underset{y_i \in \mathbb{Z}_i^k}{\operatorname{argmin}} L_{\rho,i}^k(y_i, z_i^{k,l}, \gamma_i^{k,l})$
11: send $\boldsymbol{w}_{ji}^{k,l+1}$ and $\gamma_{i,\boldsymbol{w}_{ji}}^{k,l}$ to all $j \in \mathcal{N}_i^{\mathrm{in}}$ and receive $\boldsymbol{w}_{ij}^{k,l+1}$ and $\gamma_{j,\boldsymbol{w}_{ij}}^{k,l}$ from all $j \in \mathcal{N}_i^{\mathrm{out}}$
12: compute average $\bar{\boldsymbol{x}}_i^{k,l+1} = \dfrac{1}{|\mathcal{N}_i^{\mathrm{out}}| + 1} \left(\boldsymbol{x}_i^{k,l+1} + \dfrac{\gamma_{i,\boldsymbol{x}_i}^{k,l}}{\rho} + \displaystyle\sum_{j \in \mathcal{N}_i^{\mathrm{out}}} \left(\boldsymbol{w}_{ij}^{k,l+1} + \dfrac{\gamma_{j,\boldsymbol{w}_{ij}}^{k,l}}{\rho} \right) \right)$
13: send average $\bar{\boldsymbol{x}}_i^{k,l+1}$ to all $j \in \mathcal{N}_i^{\mathrm{out}}$ and receive average $\bar{\boldsymbol{x}}_j^{k,l+1}$ from all $j \in \mathcal{N}_i^{\mathrm{in}}$
14: set $\bar{\boldsymbol{u}}_i^{k,l+1} = \boldsymbol{u}_i^{k,l+1}$, $\bar{\boldsymbol{w}}_{ji}^{k,l+1} = \bar{\boldsymbol{x}}_j^{k,l+1}$ for all $j \in \mathcal{N}_i^{\mathrm{in}}$, and $\bar{s}_i^{k,l+1} = s_i^{k,l+1}$
15: form $z_i^{k,l+1} = (\bar{\boldsymbol{x}}_i^{k,l+1}, \bar{\boldsymbol{u}}_i^{k,l+1}, \bar{\boldsymbol{w}}_i^{k,l+1}, \bar{s}_i^{k,l+1})$
16: $\gamma_i^{k,l+1} = \gamma_i^{k,l} + \rho(y_i^{k,l+1} - z_i^{k,l+1})$
17: **end for**
18: $z_i^{k+1} = z_i^{k,l_{\max}}$, $\gamma_i^{k+1} = \gamma_i^{k,l_{\max}}$
19: **end for**
20: extract $u_i[1]$ from $z_i^{k_{\max}}(t)$
21: set $u_i(t+1) = u_i[1]$ to compensate delay
22: **end for**

averaging Step 12 in Algorithm 5.2 averages primal and dual variables and thus solves the coordination QP in ADMM for any $\gamma \in \mathbb{R}^n$. Hence, Step 12 returns the correct z-update even if the iteration counters k, l differ among subsystems, i.e., if synchronization among subsystems is lost. The price to pay for allowing asynchronous execution is the additional communication of the Lagrange multipliers $\gamma_{i,\boldsymbol{w}_{ji}}^{k,l}$ for all $i \in \mathcal{S}$ in Step 11. Fortunately, this only increases the size but not the number of messages that are exchanged between subsystems.

As for the experiments with mobile robots, we here tune the penalty parameter ρ for fast convergence in simulation to an OCP minimizer found by ipopt and obtain $\rho = 4$. Moreover, we compensate any computational delay below the control sampling interval by solving OCP (5.4) for the control input $u[1]$ to be applied in the next control step.

5.3.2 Experimental Setup

Figure 5.2b depicts one of the hovercraft used in the experiments at the Automatic Control Laboratory, EPFL, Switzerland. The design combines a 150 mm diameter foam base with racing drone hardware and is light enough to float above an air hockey table. Six propellers provide thrust in each direction and result in holonomic kinematics in the 2D plane. The hovercraft pose (p_i, φ_i) is measured during experiments using an Optitrack system and a detailed description of an earlier hovercraft design as well as a deep learning approach for identifying the complex actuator dynamics is presented in [Schwan et al. 2024]. Compared to [Schwan et al. 2024], the design in Figure 5.2b contains several improvements. With respect to the experiments discussed in this chapter, the major enhancement consists of a Radxa Zero 3W which is installed onboard the hovercraft, allowing to run the EKFs and DMPC controllers onboard the hovercraft as shown in Figure 5.9b.

Alternatively, the dRTI algorithm can run offboard on workstation PCs. As indicated in Figure 5.9a, the setup is equipped with two Minisforum UM790 Pro PCs to host the DMPC and EKF applications. By assigning subsystems one and three to PC one and subsystems two and four to PC two, we ensure that dRTI messages in the offboard experiments are always transmitted to neighbors via Ethernet. That is, even though we assign multiple subsystems to the same computer, we can test the DMPC scheme in a distributed computing environment.

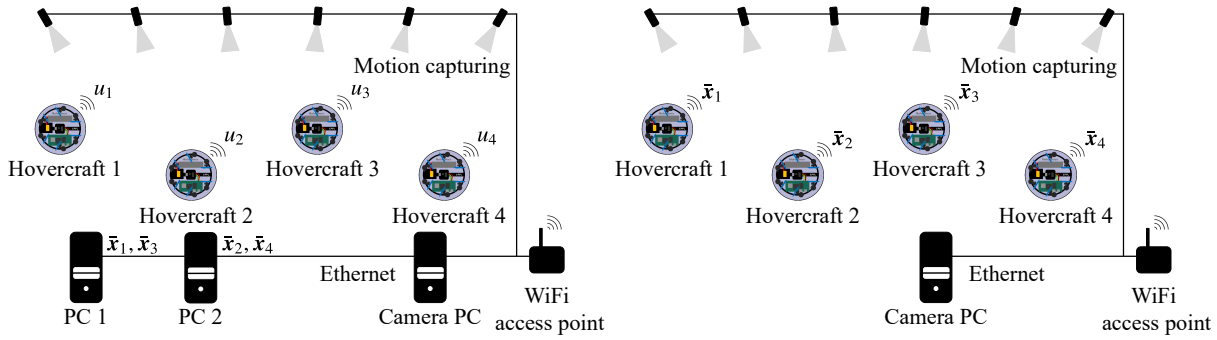
Communication in both the offboard and onboard experiments happens via the Robot Operating System (ROS2) [Macenski et al. 2022]. We use the reliable transport protocol provided by ROS2, because the best effort approach dropped many messages. For offboard experiments, messages with pose measurements and optimizer iterates are communicated via Ethernet whereas DMPC control inputs are sent via WiFi. For onboard experiments, all communication is wireless, including the pose measurements and dRTI iterates. Appendix B.2 details further implementation aspects and our open-source C++ implementation is available online.[3]

5.3.3 Experimental Results: Embedded DMPC

The dRTI scheme is tested in three different point-to-point transition scenarios and each scenario is completed with onboard and offboard DMPC execution. Video recordings are available online as a supplement to the conference paper [Stomberg et al. 2025c].[4] Table 5.5 summarizes the settings, control performance, and optimizer execution times. The first two scenarios span $T_f = 120\,\text{s}$, are without obstacle, and use a control sampling interval of $\delta = 50\,\text{ms}$ and $\delta = 150\,\text{ms}$, respectively. With the exception of the control sampling interval and number of optimizer iterations, both scenarios are equivalent. They include formation changes from a line into a rectangle and, similar to the experiments with mobile robots, a position change between neighboring hovercraft, see the top two rows in Figure 5.10. By comparing different combinations of sampling interval and optimizer iterations, i.e., $\delta = 50\,\text{ms}$ and $l_{\max} = 2$ versus $\delta = 150\,\text{ms}$ and $l_{\max} = 6$, we can investigate the

[3] https://github.com/PREDICT-EPFL/holohover

[4] https://www.youtube.com/watch?v=-ojLJUFMong



(a) Offboard DMPC execution on PCs. Optimizer iterates are communicated via Ethernet and control inputs are sent to the hovercraft via WiFi.

(b) Onboard DMPC execution on the hovercraft's Radxa boards. Position measurements and optimizer iterates are sent wirelessly.

Figure 5.9: Hardware setup for the formation control of four hovercraft on an air hockey table. The DMPC controllers run either offboard (left) or onboard (right). Hovercraft graphic by Roland Schwan [Stomberg et al. 2025c].

so-called *real-time dilemma* of MPC [Diehl et al. 2002b; Gros et al. 2020]: Choosing fast control sampling with suboptimal feedback versus slow control sampling with optimal feedback based on outdated information. As columns two and four of Table 5.5 show, faster sampling with less ADMM iterations here results in more collisions but better tracking accuracy, quantified by the averaged closed-loop cost J_{cl} (4.41). The worse tracking performance for $\delta = 150\,\mathrm{ms}$ likely results from disturbances and model-plant mismatch.

The third scenario includes an obstacle which moves in a circle near the table center as shown in the bottom row of Figure 5.10 and in Figure 5.11. Meanwhile, the cooperative hovercraft cross the table from left to right and back while avoiding collisions. Since the hovercraft are only aware of the current obstacle position when solving the OCP, $\delta = 150\,\mathrm{ms}$ proved insufficient for adapting to the obstacle motion. The plot at $t_{\mathrm{c}} = 0.50\,\mathrm{s}$ in the bottom row of Figure 5.10 illustrates the consensus error due to the low number of ADMM iterations: While the hovercraft should ideally stay in a vertical line formation, the predicted trajectories of subsystems further down the coupling graph make less progress towards the setpoint.

Figure 5.12 summarizes the dRTI exeuction times for the three scenarios with embedded DMPC. The times refer to the total durations spent on the respective dRTI step per control sampling interval. For instance, the plot on the left of the figure shows that the median time for solving the subsystem QP twice per hovercraft and per control step is approximately 11 ms in the first scenario. The plots show that dRTI executes mostly in real time, with the exception of outliers in the scenarios one and three. Most of the time is spent on wireless communication or waiting, which can also be seen from Table 5.5. In each control step, the derivative evaluation in Step 7 took at most 3.4 ms per subsystem for all optimizer iterations combined and is thus omitted from the plot.

Table 5.5: Performance statistics for hovercraft experiments. The table extends [Stomberg et al. 2025b, Table 1].

	Onboard			Offboard		
Obstacle	no	no	yes	no	no	yes
Horizon N	20	7	20	20	7	20
Sampling interval δ [ms]	50	150	50	50	150	50
$k_{\max}$	1	1	1	1	3	1
$l_{\max}$	2	6	2	30	30	30
# decision variables n	1504	568	1548	1504	568	1548
T_f	120 s	120 s	130 s	120 s	120 s	130 s
J_{cl}	8.102	14.07	11.50	4.737	22.21	5.096
# constraint violations/t_n	0.065	0.050	0.0773	0.039	0.055	0.039
# collisions	3	1	19	0	2	0
Timely MPC steps	99.50 %	100 %	98.62,%	100 %	100 %	100 %
Async. $\boldsymbol{w}$ comm. steps	0.69 %	6.19 %	6.50 %	0 %	0 %	0 %
Async. $\bar{\boldsymbol{x}}$ comm. steps	1.61 %	8.41 %	16.83 %	0 %	0 %	0 %
Time spent computing	39.32 %	13.34 %	37.79 %	60.16 %	47.25 %	62.79 %
Time spent waiting/comm.	60.68 %	86.66 %	62.21 %	39.84 %	52.75 %	37.21 %

5.3.4 Experimental Results: Offboard DMPC

The three scenarios are repeated with offboard computation which allows for more optimizer iterations as summarized in Table 5.5. The resulting performance improves and collisions are avoided for a control sampling interval of $\delta = 50\,\text{ms}$. However, the performance does not improve for $\delta = 150\,\text{ms}$, which confirms that the challenging hovercraft dynamics require fast control sampling. Figure 5.13 shows one of the maneuvers in scenario three, where the obstacle moves in a circle and the hovercraft cross the table from left to right. While the DMPC scheme successfully avoids a collision, the plot on the top right shows that the scheme struggles to accuarately track the desired yaw angle. On the one hand, this is due to the cost matrices Q_{ij} which penalize yaw differences $\varphi_i - \varphi_j$ between neighbors $i, j \in \mathcal{S}$. This choice fosters cooperation in the hovercraft orientation, but also propagates disturbances in the yaw angle to neighbors. On the other hand, these disturbances presumably result from model-plant mismatch and $\delta = 50\,\text{ms}$ appears to be insufficient for higher accuracy tracking. Less oscillations in the yaw angle occur for the obstacle whose linear-quadratic regulator uses a control sampling interval of 10 Hz.

The plots in the bottom row of Figure 5.13 further show, for all $i \in \mathcal{S}$, the distances d_{neighbor} to the next hovercraft $i + 1 \cap \mathcal{S}$, d_{obstacle} to the obstacle, and d_{boundary} to the edge of the table. The soft state constraints are violated when the hovercraft dodge the obstacle at $t_c \approx 3$ and $t_c \approx 13\,\text{s}$, but collisions are avoided thanks to the additional clearance.

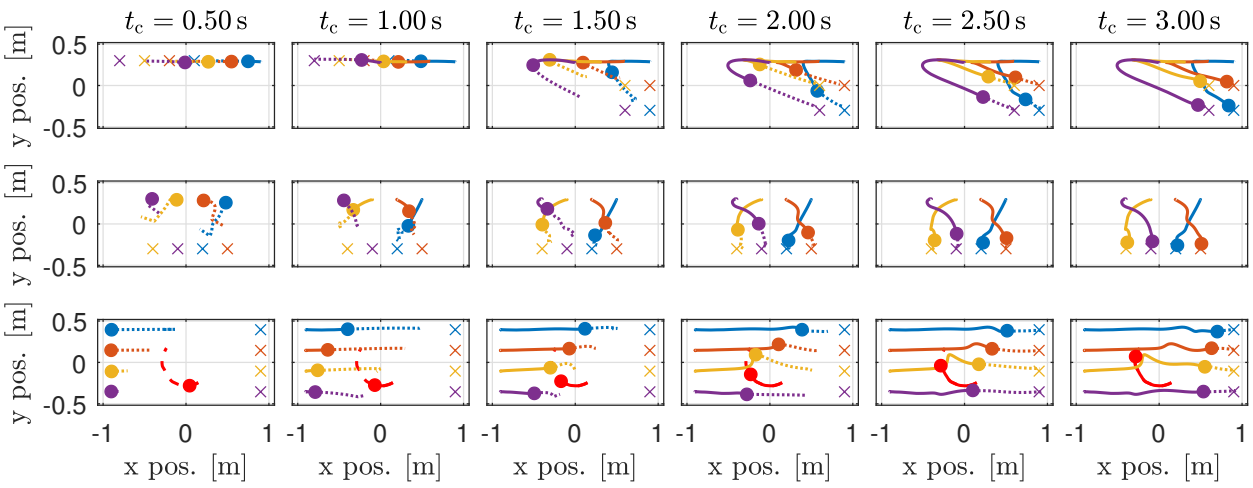


Figure 5.10: Snapshots of three maneuvers with onboard DMPC and $\delta = 50\,\mathrm{ms}$. The y- and x-axis labels on the left and bottom apply to all plots. The time stamps on the top refer to all plots within one column. The borders of each plot coincide with the boundaries of the air hockey table and circles mark the hovercraft at their current position. The cooperating hovercraft one to four are shown in blue, orange, yellow, and purple. The obstacle hovercraft is shown in red. Crosses mark the position set points p_i^{d}, dotted lines show the predicted trajectories $\bar{\boldsymbol{p}}_i$, and solid lines mark closed-loop trajectories. The future obstacle trajectory is shown by the dashed line, but is unknown to the cooperative hovercraft. Figure adapted from [Stomberg et al. 2025b].

The execution time statistics are summarized by Figure 5.14, demonstrating the real-time capabilities with solve times below the control sampling interval. In contrast to the embedded DMPC execution, communication is not a clear bottleneck when using Ethernet, similar to the mobile robots experiments in Section 5.2. The time required to evaluate derivatives via CasADi in Step 7 is negligible with a maximum of 0.1 ms per control step.

5.4 Communication Requirements and Related Work

Before discussing the relation to further results from the literature, we remark that no centralized coordinator was used in the experiments and that each subsystem ran an individual DMPC control application. Moreover, recall that Algorithms 5.1 and 5.2 rely on the reformulation of the cooperative OCP into a partially separable NLP like (3.1) via trajectory copies to decentralize ADMM. Specifically, we adopted the reformulation illustrated in Example 2.1 such that ADMM requires two communication rounds per iterations, i.e., neighbors communicate once before and once after updating the averaged state trajectory $\bar{x}^{k,l+1}$. Note that different reformulations can be employed to rewrite OCPs (5.2) and (5.4) as partially separable NLPs. As discussed in Section 2.2.3, one can also introduce even more state copies such that the two-block ADMM variant considered in

Figure 5.11: Camera images of a point-to-point transition with a dynamic obstacle and embedded DMPC. Solid lines with dotted markers are predicted trajectories and lines without markers show the latest closed-loop trajectories. Pictures by Roland Schwan.

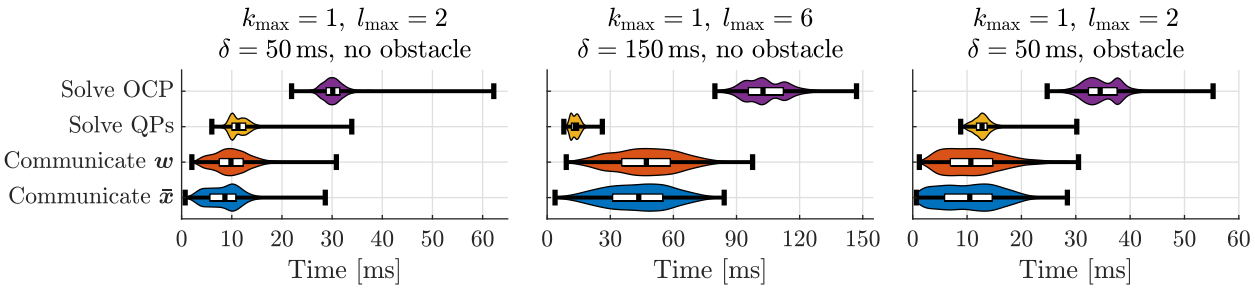


Figure 5.12: Onboard DMPC execution times per control step. Vertical lines show the minimum, lower quartile, median, upper quartile, and maximum. Colored areas display probability densities and the y-axis labels on the left apply to all plots. Figure produced with daviolinplot [Karvelis 2024] and adapted from [Stomberg et al. 2025b].

this thesis can be implemented with only one communication round per iteration, i.e., neighbors communicate once and then compute averages for their own and neighboring copy trajectories. The price to pay, however, is that this increases the NLP dimension compared to the reformulation adopted in this chapter.

Related work Section 1.2.3 provided a detailed timeline of related experiments on cooperative DMPC and we here highlight key differences to the results of this chapter. The most similar prior experiments to our validation of dRTI in terms of optimization algorithm, computing environment, and system to be controlled are reported in [Van Parys and Pipeleers 2017]. There, cooperative nonlinear DMPC of mobile robots with collision avoidance is considered based on ADMM with one iteration per 200 ms sampling interval. The computations were performed on Raspberry Pis that were mounted onboard the robots such that communication happened wirelessly. Nonetheless, solving NLPs in each ADMM iteration via ipopt was found to be the bottleneck which prevented more optimizer iterations or faster sampling. The small per-subsystem computation footprint of dSQP alleviates this bottleneck and thus enabled the fast sampling of 20 Hz for controlling the hovercraft, even with onboard execution. To the best of our knowledge, only the DMPC scheme based on Jacobi iterations in [Ebel and Eberhard 2021] achieved equally fast control sampling,

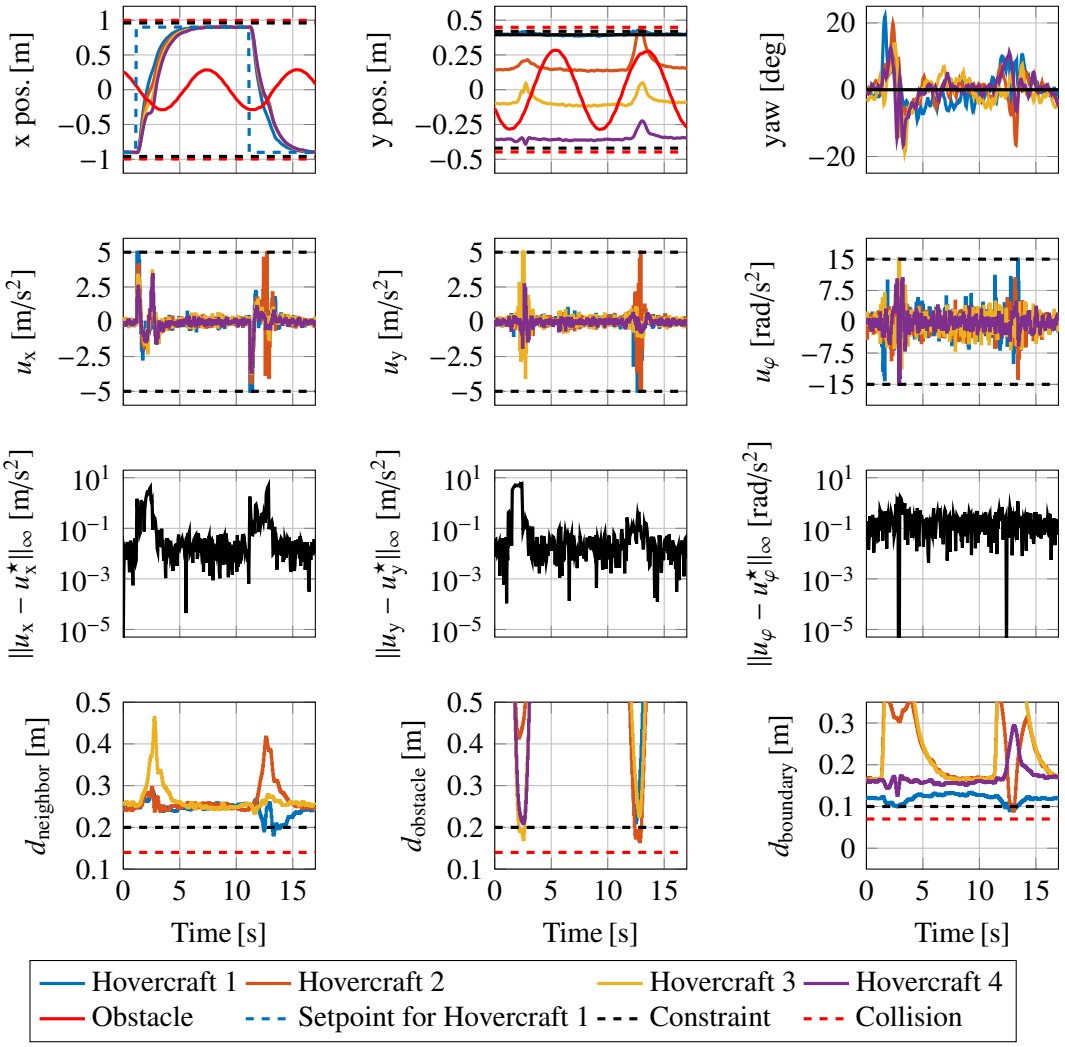


Figure 5.13: Closed-loop trajectories and optimizer residual for a point-to-point transition with dynamic obstacle, offboard execution, and $\delta = 50\,\text{ms}$. The x-axis labels and legend on the bottom apply to all plots.

albeit with centralized computation.

Combined with the theoretical analysis of Chapter 4, the experiments presented in this chapter imply that, to the best of our knowledge, dRTI is the first cooperative DMPC scheme with stability guarantees *and* practical validation that does not require centralized coordination or feasible warm starts. Notably, Sensitivity-Based DMPC (SB-DMPC) also enjoys stability guarantees and has been validated in hardware experiments [Pierer von Esch et al. 2025a]. A comparison of SB-DMPC and dRTI from a theoretical point of view is given in Section 4.7. Here, we compare the experimental validation of SB-DMPC applied to magnetic levitation movers in [Pierer von Esch et al. 2025b] to our earlier results [Stomberg et al. 2023] and [Stomberg et al. 2025c]. As for the mobile robot experiments in Section 5.2, the experiments with SB-DMPC use a sampling interval of $\delta = 200\,\text{ms}$ and rely on Ethernet for neighbor-to-neighbor communication. For the experiments,

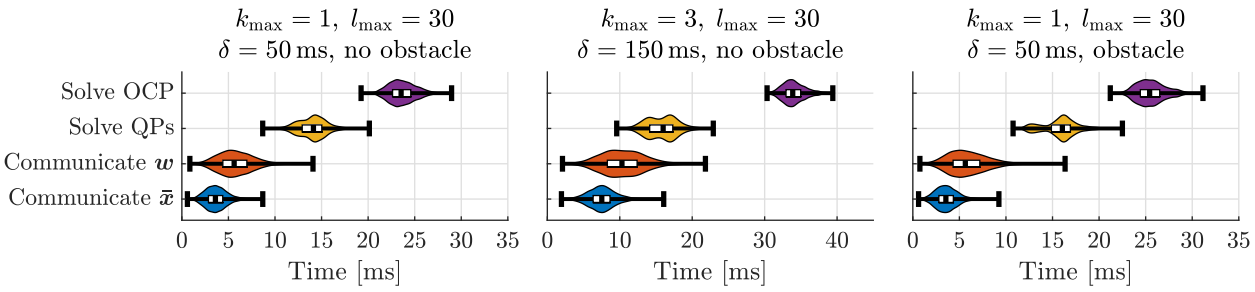


Figure 5.14: Offboard DMPC execution times per control step. Figure produced with daviolinplot [Karvelis 2024] and adapted from [Stomberg et al. 2025b].

SB-DMPC ran on Raspberry Pis whose computational power is closer to the Radxas installed onboard the hovercraft than to workstation PCs. The magnetic levitation movers are equipped with low-level PID controllers such that the movers are modeled as double integrator dynamics in the OCP, similarly to the hovercraft in Section 5.3. An important advantage of the SB-DMPC experiments is that they contain scenarios where the movers are physically connected via springs, i.e., some of the SB-DMPC experiments feature dynamically coupled subsystems whereas this chapter exclusively discusses decoupled dynamics. In contrast, an advantage of the dRTI hovercraft experiments is that they consider much faster control sampling intervals of 50 ms *and* wireless communication. Note that the specifications of each experiment, i.e. wired vs. wireless communication or coupled vs. decoupled dynamics, depends on the available laboratory equipment. Thus, the above comparisons refer to differences in the experiments, but not to inherent strengths or weaknesses of dRTI and SB-DMPC.

5.5 Summary

This chapter has presented experimental results for dRTI applied to robot formation control. Distributed hardware setups allowed to test cooperative DMPC in real time, both on embedded systems and offboard workstations. In particular, the dRTI scheme facilitated control sampling intervals between 50 ms and 200 ms and solved non-convex state constraints, recovering established collision avoidance strategies by linearizing minimum-distance constraints on the SQP layer. While the DMPC scheme proved effective accross a wide range of scenarios, the experiments also highlight current limitations of the approach. First, the cooperative DMPC scheme could not attain the fast sampling frequencies possible with classical decentralized control methods. While this is to be expected of most distributed control schemes subject to communication delays, the applicability to systems which require sampling frequencies of 100 Hz or faster remains challenging. Second, the early termination of ADMM limits the consensus between subsystems, causing violations in coupled constraints. However, these limitations are not fundamental drawbacks of dRTI, as can be seen from the theoretical guarantees provided in Chapter 4. In fact, future implementations on faster hardware may yield even better performance, also for fast mechatronic systems.

6 Computational Performance for Large-Scale Optimal Control

The previous chapters discuss the dRTI framework for DMPC and its application to robot formation control. This chapter investigates the numerical performance of dRTI for large-scale systems on the example of frequency control for power networks. By considering networks of increasing size, we test the scalability for linear-quadratic DMPC using ADMM and for nonlinear DMPC using dSQP. On the one hand, we analyze the change in solver performance if new subsystems are added to a network. On the other hand, we test state-of-the-art centralized solvers to contrast the performance of DMPC and centralized MPC. The results of this chapter have appeared in the conference paper [Stomberg et al. 2025b].

6.1 Power Network Benchmark and Implementation

We consider meshed power networks as depicted in Figure 6.1 and note that a similar system serves as benchmark for linear DMPC based on system level synthesis in [Alonso et al. 2023]. Here, we additionally consider nonlinear system dynamics and we use parameters similar to [de Jong and Lazar 2023].

Power Network Model

Each network is described by a bus index set $\mathcal{D}$ which can be partitioned into a set of synchronous generators $\mathcal{P}$ and a set of uncontrollable loads $\mathcal{L}$. The state of bus $n \in \mathcal{D}$ is $q_n \doteq (\theta_n, \omega_n) \in \mathbb{R}^2$, where θ_n denotes the voltage angle and ω_n is the angular velocity. The state is measured relative to a synchronous equilibrium, i.e., if $\omega_n = 0$, then bus n is synchronous to the nominal grid frequency of 50 Hz. The considered control task is to synchronize all buses at the nominal frequency, which corresponds to an output regulation problem where the system output ω_n is to be steered to the origin for all $n \in \mathcal{C}$. Each bus $n \in \mathcal{D}$ is subject to the nonlinear synchronous machine dynamics [Dörfler and Bullo 2012]

$$\begin{aligned} \begin{bmatrix} \dot{\theta}_n(t_\mathrm{c}) \\ M_n \dot{\omega}_n(t_\mathrm{c}) \end{bmatrix} &= \begin{bmatrix} \omega_n(t_\mathrm{c}) \\ -D_n \omega_n(t_\mathrm{c}) + P_n(t_\mathrm{c}) + p_n(t_\mathrm{c}) + w_n(t_\mathrm{c}) \end{bmatrix} \\ P_n(t_\mathrm{c}) &\doteq - \sum_{m \in \mathcal{M}_n} a_{nm} \sin\left(\theta_n(t_\mathrm{c}) - \theta_m(t_\mathrm{c})\right). \end{aligned} \tag{6.1}$$

For all $n \in \mathcal{D}$, the inertia and damping constants are denoted by $M_n, D_n > 0$, P_n is the power transfer with neighboring buses, and $\mathcal{M}_n \doteq \{m \in \mathcal{D} \mid a_{nm} > 0\}$ is the set of neighbors. The coupling weights $a_{nm} \geq 0$ are symmetric such that $a_{nm} = a_{mn}$ for all $n, m \in \mathcal{D}$. For all generators $n \in \mathcal{P}$, $p_n \in [-0.3, 0.3]$ pu is the controllable power injection and $w_n \in \mathbb{R}$ is the uncontrollable load for

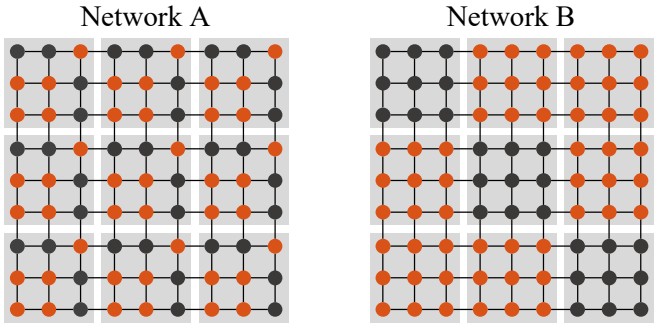


Figure 6.1: Power networks with 81 buses. Figure adapted from [Stomberg et al. 2025b].

all $n \in \mathcal{L}$. We assume that each bus is either a generator or a load and thus we set $w_n = 0$ for all $n \in \mathcal{P}$ and $p_n = 0$ for all $n \in \mathcal{L}$. In addition to the above generator constraints, the state of bus $n \in \mathcal{D}$ is constrained to the set [de Jong and Lazar 2023]

$$\mathbb{Q}_n \doteq \left\{ q_n \in \mathbb{R}^2 \;\middle|\; \begin{array}{l} -1.6\,\pi/\mathrm{s} \leq \omega_n \leq 1.6\,\pi/\mathrm{s} \\ -\dfrac{\pi}{2} \leq \theta_n - \theta_m \leq \dfrac{\pi}{2} \quad \forall m \in \mathcal{M}_n \end{array} \right\}. \tag{6.2}$$

Besides nonlinear DMPC via dSQP, the scalability analysis to be presented below also considers linear DMPC via ADMM. The latter employs a linearization of the nonlinear dynamics (6.1), where the power transfer with neighboring buses is replaced by $P_n^{\mathrm{lin}}(t) \doteq -\sum_{m \in \mathcal{M}_n} a_{nm}(\theta_n(t) - \theta_m(t))$.

Optimal Control Problem Design

To obtain a distributed control scheme, we partition the network by assigning each bus $n \in \mathcal{D}$ to one subsystem $i \in \mathcal{S}$. There exists a range of possible strategies for designing the coupling graph topology associated with the set of subsystems $\mathcal{S}$ [Chanfreut et al. 2021b]. Here, we consider partitions of square networks into square subsystems as shown in Figure 6.1. This choice allows to easily scale up the overall network size $|\mathcal{D}|$ by adding more subsystems and thus new buses to the network. We denote the set of buses assigned to subsystem $i \in \mathcal{S}$ by $\mathcal{D}_i \subseteq \mathcal{D}$ and we assume that

$$\mathcal{D} = \bigcup_{i \in \mathcal{S}} \mathcal{D}_i \quad \text{and} \quad \mathcal{D}_i \cap \mathcal{D}_j = \emptyset \quad \text{for all} \quad i, j \in \mathcal{S} \quad \text{if} \quad i \neq j.$$

The sets of generators and loads assigned to subsystem $i \in \mathcal{S}$ are denoted by $\mathcal{P}_i$ and $\mathcal{L}_i$, respectively. For all $i \in \mathcal{S}$, the subsystem state is $x_i \doteq (q_n)_{\mathcal{D}_i}$, the input is $u_i \doteq (p_n)_{n \in \mathcal{P}_i}$, and the uncontrollable loads are modeled as disturbance $d_i \doteq (w_n)_{n \in \mathcal{L}_i}$. The DMPC scheme considers coupling based on the in-neighbors

$$\mathcal{N}_i^{\mathrm{in}} \doteq \left\{ j \in \mathcal{S} \;\middle|\; \exists (n, m) \in \mathcal{D}_i \times \mathcal{D}_j \text{ such that } a_{nm} > 0 \right\} \quad \text{for all} \quad i \in \mathcal{S}.$$

We denote the set of coupled buses in neighboring subsystems by $\mathcal{M}_i^{\mathrm{in}} \doteq \cup_{n \in \mathcal{N}_i} \mathcal{M}_n \setminus \mathcal{D}_i$. The neighboring state of subsystem $i \in \mathcal{S}$ then is $x_{\mathcal{N}_i^{\mathrm{in}}} \doteq (\theta_m)_{m \in \mathcal{M}_i^{\mathrm{in}}}$. That is, as for the inverted pendulum

example in Chapter 4, $x_{\mathcal{N}_i^{\text{in}}}$ only collects those states in $(x_j)_{j\in\mathcal{N}_i^{\text{in}}}$ which directly affect subsystem i in order to avoid unnecessary decision variables in the OCP.

For all $i \in \mathcal{S}$, the continuous-time dynamics in abstract form read

$$\dot{x}_i(t_c) = f_i^c(x_i(t_c), x_{\mathcal{N}_i^{\text{in}}}(t_c), u_i(t_c), d_i(t_c)), \quad x_i(0) = x_{i,0}, \tag{6.3}$$

where $f_i^c : \mathbb{R}^{n_{x_i}} \times \mathbb{R}^{n_{x_i}^{\text{in}}} \times \mathbb{R}^{n_{u_i}} \times \mathbb{R}^{n_{d_i}} \to \mathbb{R}^{n_{x_i}}$. In contrast to Section 4.1, the dynamics (6.3) include disturbances d_i for all $i \in \mathcal{S}$ which are treated as parameters in the OCP. For simplicity, the disturbance predictions over the horzon are set equal to the current disturbance $d_i(t)$ which is assumed to be known. An alternative that could improve control performance would be to forecast loads [Hong et al. 2020]. We discretize continuous-time dynamics with a zero-order hold on u_i, $x_{\mathcal{N}_i^{\text{in}}}$, and d_i using the explicit second-order Heun method. The integration step size is equal to the control sampling interval δ and the zero-order hold on $x_{\mathcal{N}_i^{\text{in}}}$ preserves the sparse coupling structure between subsystems at the cost of a reduced integration accuracy, similar to the inverted pendulum example in Chapter 4. For all $i \in \mathcal{S}$, the state constraint sets $\mathbb{X}_i$ and $\mathbb{X}_{ij}$, $j \in \mathcal{N}_i^{\text{in}}$ in OCP (4.5) include the bus constraint sets $\mathbb{Q}_n$ for all $n \in \mathcal{D}_i$. Specifically, the box constraints on ω_n and the angle differences $\theta_n - \theta_m$ for all $n, m \in \mathcal{D}_i$ in (6.2) are included in $\mathbb{X}_i$ and the phase difference constraints $\theta_n - \theta_m$ with respect to coupled buses $m \in \mathcal{D}_j$ are included in $\mathbb{X}_{ij}$, where $j \in \mathcal{N}_i^{\text{in}}$ and $i \in \mathcal{S}$. For all $i \in \mathcal{S}$, the generator box constraints translate into an input box constraint set $\mathbb{U}_i$.

The control sampling interval $\delta = 0.1\,\text{s}$, discrete-time OCP horizon $N = 100$, coupling weights $a_{nm} = 0.2$ if applicable, and nominal values $M = 0.167\,\text{pu}\cdot\text{s}^2$ and $D = 0.0045\,\text{pu}\cdot\text{s}$ are chosen similar to [de Jong and Lazar 2023]. For all $i \in \mathcal{S}$, we design OCP (4.5) with quadratic costs

$$\ell_i(x_i, u_i) \doteq \sum_{n\in\mathcal{D}_i} \frac{1}{2}\left(q_n^\top Q q_n + p_n^\top R p_n\right) \quad \text{and} \quad V_{\text{f}_i} \doteq \sum_{n\in\mathcal{D}_i} \frac{1}{2} q_n^\top P q_n$$

with $Q = \text{diag}(0, 1)$, $R = 0.1$, and $P = Q$. The scaling parameter for the terminal penalty in OCP (4.5) is chosen as $\beta = 1$.

To apply ADMM or dSQP, we reformulate OCP (4.5) as a partially separable NLP like (3.1) by introducing copies for the neighboring states $x_{\mathcal{D}_i^{\text{in}}}$, cf. Example 2.1. Moreover, we regularize the OCP by adding the quadratic penalty $cz_i^\top z_i$ with weight $c = 10^{-4}$ to the cost function f_i for all $i \in \mathcal{S}$, which ensures that the GN Hessian approximations $H_i = \nabla^2_{z_i z_i} f_i$ are positive definite.

Implementational Details

To evaluate the performance of ADMM and dSQP for the considered power networks, we have implemented multi-threaded versions of both methods in Julia v. 1.10.5. The source code to run the simulations is available online.[1] All simulations are carried out on a Debian Linux Virtual Machine (VM). The VM ran on a server at TU Dortmund University which is equipped with two AMD EPYC 7742 64-core processor sockets and 1 TB RAM of which the VM can access 40 CPU

[1] `https://github.com/OptCon/dmpc_scalability`

cores and 64 GM RAM. The server did not execute any further jobs during the simulations to avoid interference on the solver time measurements. For centralized MPC, we test centralized solvers in combination with parallel linear algebra packages. Thus, both distributed MPC and centralized MPC benefit from the multi-core CPU architecture in our implementation.

For solving QPs in centralized linear-quadratic MPC, we test CPLEX v. 22.1.0 and OSQP v. 0.8.1, subsequently referred to as centralized OSQP [IBM 2024; Stellato et al. 2020]. We interface CPLEX via JuMP v. 1.23.2 which we use to construct a centralized QP model [Lubin et al. 2023]. For centralized OSQP, we use Intel oneMKL Pardiso as a parallel linear algebra package [Intel 2024; Schenk et al. 2001]. For linear-quadratic DMPC, we use ADMM as given in Algorithm 4.1. To accelerate the execution of ADMM, we parallelize the solution of the subsystem QPs in Step 3 and the dual update in Step 5 of Algorithm 4.1. The averaging Step 4 is implemented as the efficient matrix-vector product

$$z^{l+1} = M_{\text{avg}} y^l,$$

where the ADMM averaging matrix M_{avg} is computed offline. The subsystem QPs are solved using OSQP running the sequential QDLDL linear algebra solver [Stellato et al. 2020].

For non-convex NLPs arising in centralized NMPC, we use the interior point solver MadNLP v. 0.8.4 in combination with Intel oneMKL Pardiso [Shin et al. 2021; Shin et al. 2024]. As decentralized method for nonlinear DMPC, we test a dSQP implementation with fixed ADMM iterations on the inner level as summarized in Algorithm 4.2. The derivatives in Step 3 of dSQP are evaluated using automatic differentiation via JuMP's MathOptInterface v. 1.32.2 [Legat et al. 2021]. The derivative computation is parallelized by assigning one thread to each subsystem and on the inner level we use the multi-threaded ADMM implementation from above.

The scalability analysis to be presented below investigates the number of iterations required by centralized OSQP, ADMM, and dSQP to obtain approximate OCP solutions. To this end, we terminate these methods based on the residual

$$r \doteq \left\| \begin{bmatrix} F_1(z_1, \nu_1, \mu_1, \lambda) \\ \vdots \\ F_s(z_S, \nu_S, \mu_S, \lambda) \\ \sum_{i \in \mathcal{S}} E_i z_i \end{bmatrix} \right\|_\infty \quad \text{with} \quad F_i(z_i, \nu_i, \mu_i, \lambda) \doteq \begin{bmatrix} \nabla_{z_i} L_i(z_i, \nu_i, \mu_i, \lambda) \\ g_i(z_i) \\ \max(0, h_i(z_i)) \\ \min(0, \mu_i) \\ \left([\mu_i]_j \cdot [h_i]_j \right)_{j \in \{1, \ldots, n_{h,i}\}} \end{bmatrix}.$$

In order to keep the overall ADMM and dSQP execution time low, we set the the OSQP tolerances in the subsystem QP solves as $\epsilon_{\text{abs}} = \epsilon_{\text{rel}} = \epsilon_{\text{dual}} = \epsilon_{\text{dual}} = \text{tol}/10$, where tol denotes the specified tolerance on r. Put differently, we allow for approximate subsystem QP solutions for the sake of reducing the OSQP execution time inside ADMM.

Table 6.1: Problem parameters for the open-loop scalability study. Table adapted from [Stomberg et al. 2025b].

Case	$\lvert\mathcal{D}_i\rvert$	$\lvert\mathcal{L}_i\rvert$	$\lvert\tilde{f}_{n,0}\rvert$	Results shown in Figure
1	9	2	32 mHz	Fig. 6.2 (QP) and Fig. 6.3 left columns (NLP)
2	16	4	32 mHz	Fig. 6.2 (QP) and Fig. 6.3 center columns (NLP)
3	25	5	32 mHz	Fig. 6.2 (QP) and Fig. 6.3 right columns (NLP)
4	9	4	32 mHz	Fig. 6.4 (NLP)
5	9	6	32 mHz	Fig. 6.4 (NLP)
6	9	2	48 mHz	Fig. 6.5 (NLP)
7	9	2	64 mHz	Fig. 6.5 (NLP)

6.2 Scalability Study

We investigate the optimizer performance for a range of OCP parameters including the number of decision variables n, the number of buses per subystem $|\mathcal{D}_i|$, the number of subsystems $|\mathcal{S}|$, the number of load buses per subsystem $|\mathcal{L}_i|$, and the initial condition for the frequency $\tilde{f} = \omega/(2\pi)$.

Case Study Design

Table 6.1 summarizes the parameter values for seven different test cases. For each test case, the number of subsystems in the network is increased from 4 to 36 such that n grows, but the remaining parameters are kept constant. The OCPs within each test case are constructed as follows. First, we sample random parameters from uniform distributions $M_n \in [0.9, 1.1]M$, $D_n \in [0.9, 1.1]D$, and $\tilde{f}_{n,0} \leq |\tilde{f}_{n,0}|$ for all $n \in \mathcal{D}_1$. Moreover, for all $n \in \mathcal{D}_1$, we decide at random whether $n \in \mathcal{P}_1$ or $n \in \mathcal{L}_1$. The initial condition for the angles is $\theta_n(0) = 0$ for all $n \in \mathcal{D}_1$. Then, we add subsystems with the same parameters as the first subsystem to obtain square networks as shown in Network A in Figure 6.1. Choosing the same parameters for all subsystems ensures that the only major change between the networks within a test case lies in the number of subsystems and thus the problem dimension n, but not the initial condition, bus parameters, or placement of loads and generators. For centralized OSQP, ADMM, and dSQP, we tune the penalty parameter ρ for each test case based on the network with four subsystems. The parameter ρ thus only depends on the test case and not on the network size.

We evaluate the optimizer performance based on the required number of iterations and the associated time for solving the OCP. For centralized OSQP, ADMM, and dSQP this refers to suboptimal solutions based on a pre-specified tolerance for the residual r defined above. In contrast, the interior point solvers CPLEX and MadNLP solve each OCP to high accuracy. Specifying low accuracies yielded little reduction in the MadNLP solve time, because fast convergence occurred primarily in the final interior point iterations. Omitting these final MadNLP iterations degraded the solution quality, but had little effect on the overall execution time.

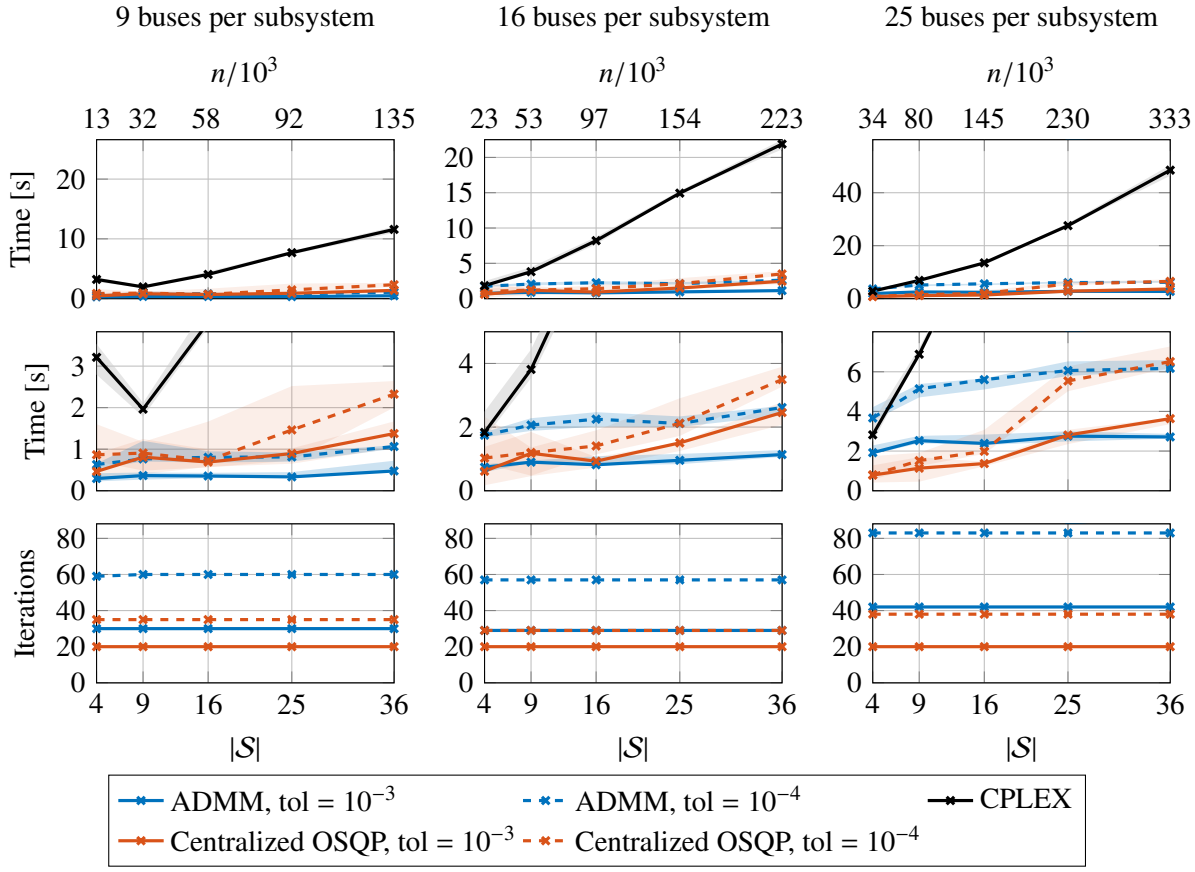


Figure 6.2: Computational performance for convex QPs with $|\mathcal{D}_i| = \{9, 16, 25\}$ buses per subsystem. Figure adapted from [Stomberg et al. 2025b].

For our custom ADMM and dSQP implementations, we first initialize all data structures prior to the time meausrements, which includes one call to OSQP for solving the subsystem QPs. The reported measurements thus cover the code that would run in an online DMPC implementation. However, the simulations run on a single server and thus DMPC communication delays that would occur in decentralized implementations are not included. All measurements are taken five times to account for runtime uncertainty.

Simulation Results

Linear-quadratic DMPC: Figure 6.2 summarizes the results where OCP (4.5) is a convex QP with the linearized power exchange P^{lin}. The legend on the bottom of the figure applies to all plots and each plot includes two x-axes. The x-axis on the top refers to the number of decision variables n in the centralized OCP, rounded to the nearest thousand, and the x-axis on the bottom indicates the number of subsystems $|\mathcal{S}|$ present in the network. For both x-axes, the axis labels on the top and bottom apply to all plots within one column. Likewise, the y-axis description on the left applies to all plots within one row. For the time measurements, crosses mark the median and shaded areas highlight the range from minimum to maximum of the five runs of each simulation and the same

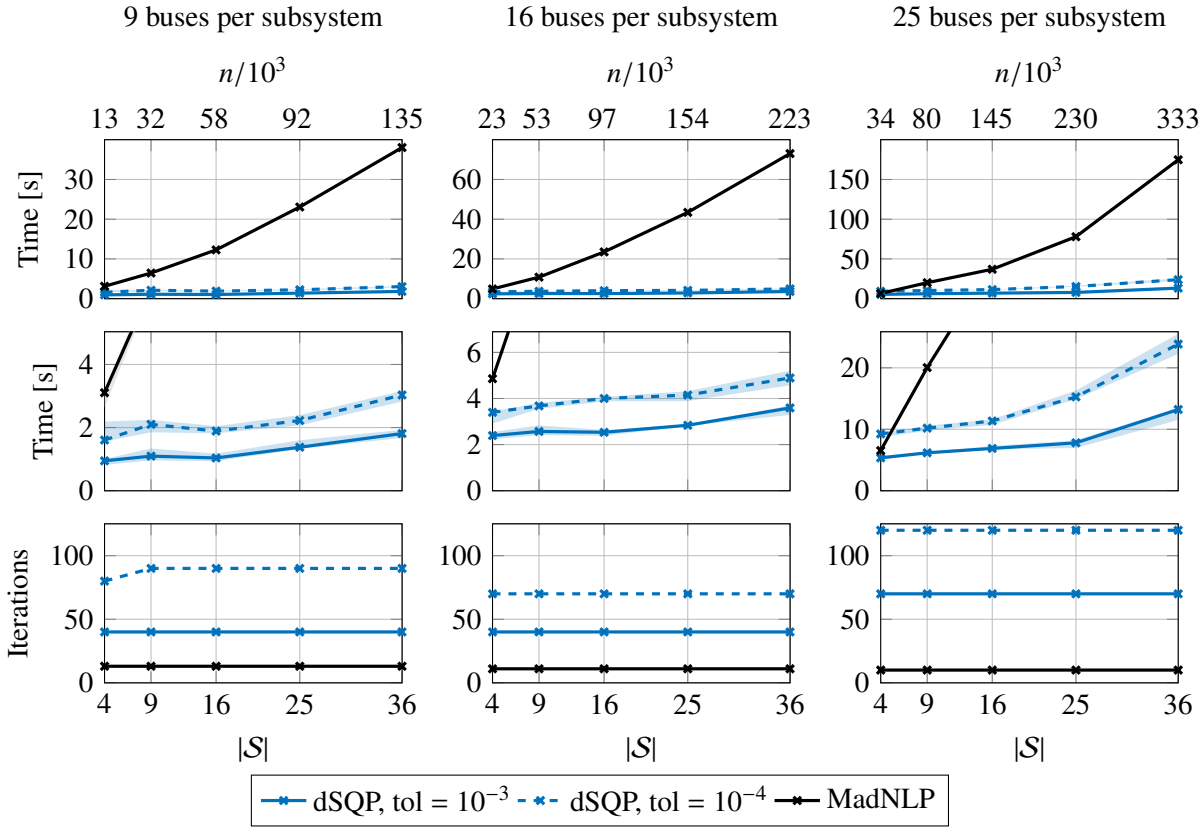


Figure 6.3: Computational performance for non-convex NLPs with $|\mathcal{D}_i| = \{9, 16, 25\}$ buses per subsystem. Figure adapted from [Stomberg et al. 2025b].

applies to all other figures in this section.

The top two rows of Figure 6.2 show the solve times for each method and the center row is an enlarged version for better visualization. CPLEX is much slower than ADMM and centralized OSQP, but also obtains high-accuracy solutions. The solve times of CPLEX on the one hand and ADMM and OSQP on the other hand are thus not directly comparable. The CPLEX results rather show that solving large-scale OCPs on multi-core CPUs is currently computationally infeasible, even with a state-of-the-art solver and parallelization. Centralized OSQP and ADMM, however, rapidly converge to suboptimal solutions and scale favorably. Notice how the required number of iterations for ADMM and OSQP depends on the problem dimension n_i per subsystem and on the desired accuracy, but not on the number of subsystems. This suggests that both methods scale favorably, as long as the per-iteration time can be kept constant through parallelization. For centralized OSQP, implementations for GPUs and FPGAs may provide even further speedup compared to the multi-core CPU architecture considered here [Schubiger et al. 2020; Wang et al. 2023].

Nonlinear DMPC: Figure 6.3 shows the effects of the subsystem size for non-convex NLPs with the nonlinear dynamics (6.1). MadNLP and dSQP require nearly constant iterations when new subsystems are added to the network, again indicating promising scalability if the per-iteration time stays constant. However, the execution times of MadNLP and dSQP are not directly comparable,

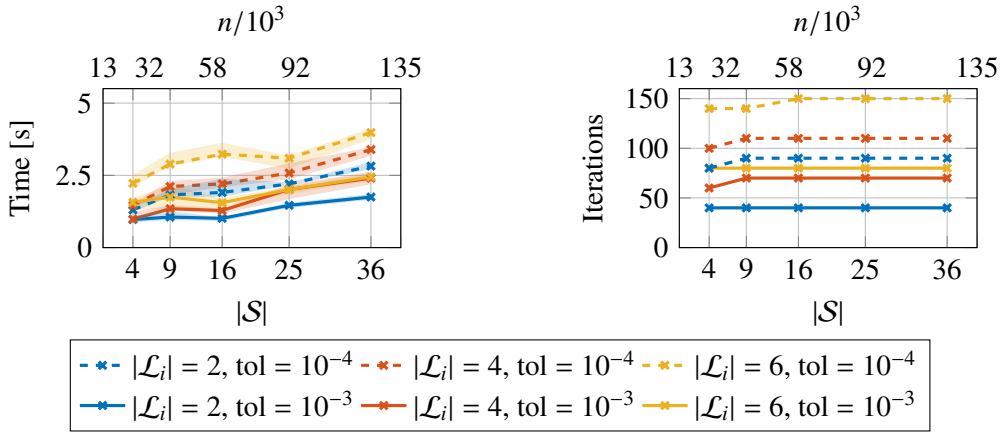


Figure 6.4: Computational peformance of dSQP for non-convex NLPs with $|\mathcal{L}_i| = \{2, 4, 6\}$ loads per subsystem. Figure adapted from [Stomberg et al. 2025b].

because MadNLP converges to higher accuracy. The results illustrate the remaining challenges of solving large-scale NLPs to optimality with parallelization on multi-core CPUs. However, parallelization on GPUs seems promising for overcoming this bottleneck of centralized NMPC [Pacaud et al. 2024; Shin et al. 2024].

So far, we have analyzed the scalability of networks with varying subsystem size, cf. Figures 6.2 and 6.3. In contrast, Figures 6.4 and 6.5 investigate the effect of different subsystem settings, but with a constant size of nine buses per subsystem. Specifically, Figure 6.4 shows what happens if generators within a subsystem are replaced by loads, which corresponds to control inputs being replaced by disturbances. The figure confirms that intuitively more challenging control problems also require more optimizer iterations to reach convergence. However, as for problems with varying subsystem size, we observe that the number of optimizer iterations is largely independent of the number of subsystems in the network. The same applies to Figure 6.5, where we increase the magnitude of the initial disturbance. Thus, the crucial observation that DMPC scales with a constant number of optimizer iterations applies to a wide range of settings, confirming the potential of dRTI for large-scale CPsoS.

Remark 6.1 (Distributed vs. centralized MPC)**.** *The results in Figures 6.2 and 6.3 do not imply that ADMM and dSQP are better suited than CPLEX and MadNLP for all OCPs in general. In fact, the results in [Kozma et al. 2015] include examples where ADMM does not compare well with centralized solvers. Instead, our results demonstrate that the graph structure in OCPs can result in a constant iteration requirement for decentralized optimization methods. This, together with the constant per-iteration execution times for ADMM observed in [Huber et al. 2022], indicates good scalability of cooperative DMPC. Centralized MPC can also benefit from this effect, because we observe constant iteration numbers for OSQP and MadNLP when subsystems are added to the network. Thus, both centralized MPC and DMPC have the potential to scale well if the per-iteration execution time can be kept low through parallelization.* □

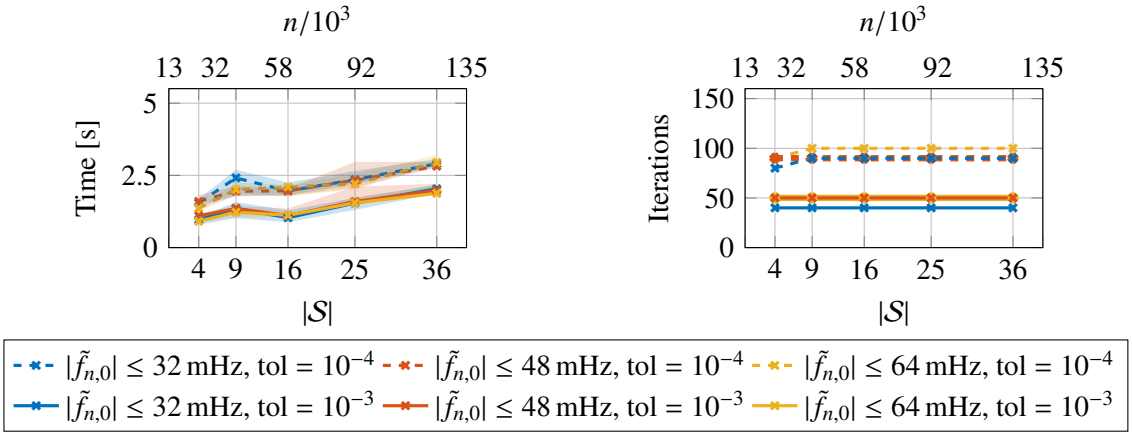


Figure 6.5: Computational peformance of dSQP for non-convex NLPs with initial frequency distrubance $|\tilde{f}_{n,0}| \leq \{32, 48, 64\}$ mHz. Figure adapted from [Stomberg et al. 2025b].

6.3 Closed-Loop Control Performance

The analysis in the previous section shows that, for the considered scenarios, the required number of ADMM and dSQP iterations for reaching suboptimal solutions is largely independent of the number of subsystems. The scalability of DMPC thus depends on the control performance that results from these suboptimal controls. This section analyzes the performance of suboptimal linear and nonlinear DMPC in simulation.

Case Study Design

We test linear-quadratic and nonlinear DMPC on the two 81-bus networks shown in Figure 6.1. The grids differ in the positioning of loads and buses. Whereas each subsystem in Network A contains loads and generators, the subsystems in Network B include either loads or generators. Network B therefore requires closer cooperation between subsystems, which allows us to test the effect of coupling constraint violations due to the early ADMM termination in each control step. Figure 6.6 illustrates the scenario, where all buses are initially synchronized. The loads then exhibit a random jump of up to -0.1 pu at $t = 0.1$ s. We quantify control performance as the averaged closed-loop cost J_{cl} defined in (4.41). We first test the ideal centralized controller where the OCP is solved to high accuracy via CPLEX (linear-quadratic MPC) or MadNLP (NMPC) and we denote the resulting averaged closed-loop cost as $J_{\text{cl}}^{\star}$. Then, we apply ADMM (linear-quadratic DMPC) or dSQP (nonlinear DMPC) to obtain the averaged closed-loop cost J_{cl}. The suboptimality due to the early termination is the relative control performance $J/J^{\star}$.

Simulation Results

The plots on the left of Figure 6.6 show the trajectories for the angles θ_n, frequency $\bar{f}_n \doteq \tilde{f} + 50$ Hz, and injected power u_n for all $n \in \mathcal{D}$. The plot on the bottom left shows both the generator inputs p_n

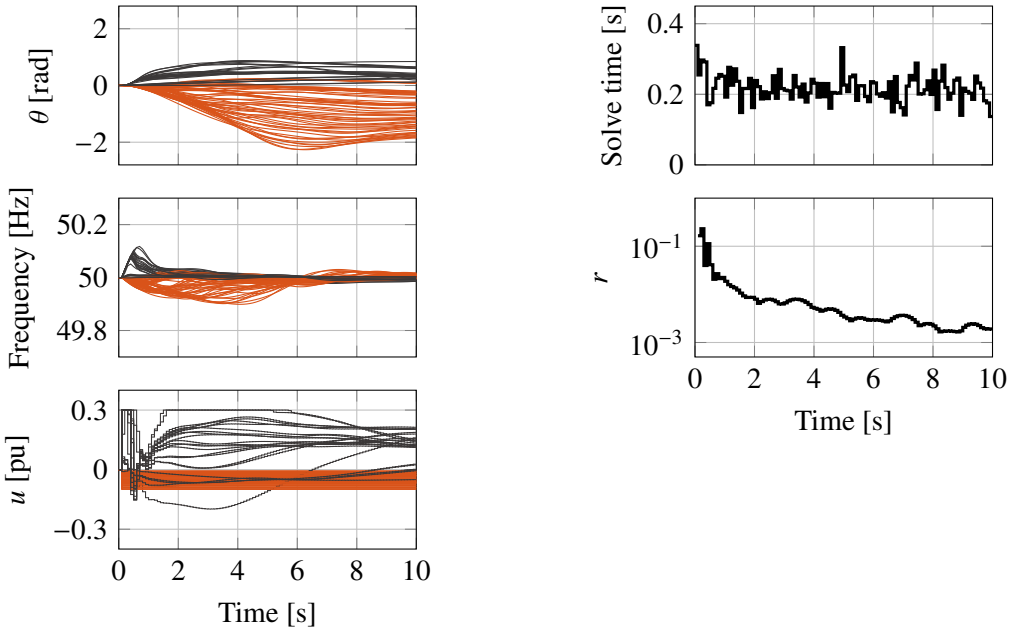


Figure 6.6: Closed-loop nonlinear DMPC simulation for Network B of Figure 6.1. The OCP for the 81-bus Network contains $n = 31815$ decision variables and is solved via dSQP with $k_{\max} = 1$ and $l_{\max} = 10$. Generator trajectories are shown in gray and load trajectories are shown in orange. Figure adapted from [Stomberg et al. 2025b].

and loads w_n for better visualization, even though the loads w_n are strictly speaking disturbances and not inputs. The simulation demonstrates the system-optimizer convergence of dRTI in closed loop: The chosen number of iterations $l_{\max} = 10$ is less than the 40 iterations required in Figure 6.4 to reach an accuracy of 10^{-3}, but dSQP synchronizes the buses thanks to the warm-starting.

Figure 6.7 summarizes the relative control performance $J_{\text{cl}}/J_{\text{cl}}^{\star}$ and solve time for ADMM and centralized OSQP, where the solve times are given as percentage of the CPLEX solve time. Shaded areas mark the span from minimum to maximum recorded solve times and crosses denote the median. The results show that few iterations yield good control performance and require a fraction of the CPLEX solve time, both for centralized MPC using OSQP and for DMPC using ADMM.

The same phenomenon can be observed in Figure 6.8 which compares nonlinear DMPC using dSQP to centralized NMPC using MadNLP. Nonlinear DMPC achieves 99 % control performance with $l_{\max} = 7$ ADMM iterations per control step, requiring less than seven percent of the MadNLP execution time. Thus, the suboptimal inputs can indeed suffice in closed-loop. In combination with the results from the previous section, this suggests that DMPC has the potential to scale well. However, the simulations also highlight a drawback of ADMM-based DMPC: For systems that require a high level of consensus, the control performance deteriorates unless more iterations are applied per control step. This can be seen from Figures 6.7 and 6.8, where the control performance obtained with few optimizer iterations is clearly worse for Network B than for Network A, because Network B requires the subsystems to cooperatively balance power supply and demand.

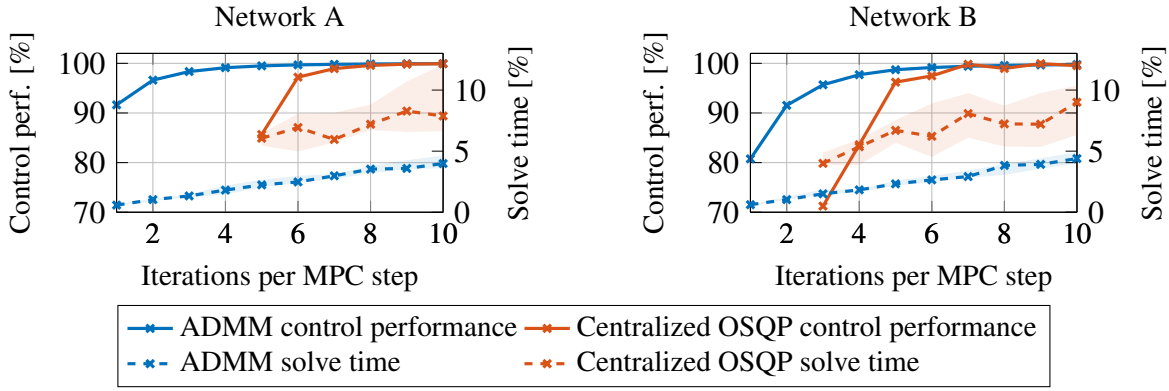


Figure 6.7: Relative control performances and solve times for DMPC (ADMM) and suboptimal centralized MPC (centralized OSQP) compared to centralized MPC using CPLEX. Figure adapted from [Stomberg et al. 2025b].

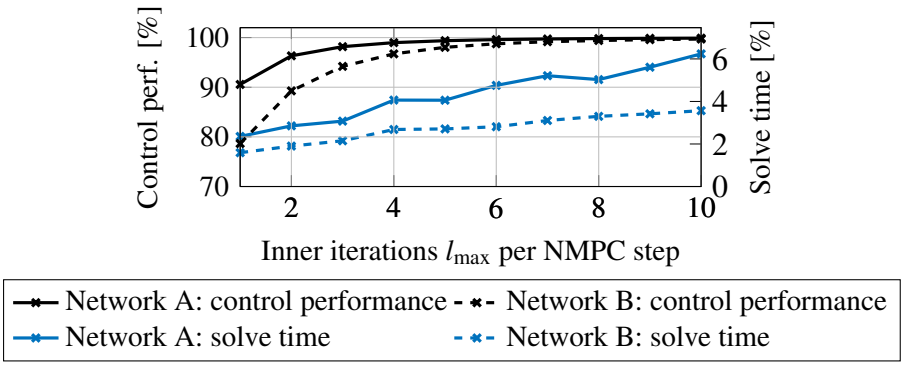


Figure 6.8: Relative control performance $J_{\text{cl}}/J_{\text{cl}}^{\star}$ and solve times for nonlinear DMPC using dSQP compared to centralized NMPC using MadNLP. Figure adapted from [Stomberg et al. 2025b].

6.4 Summary

We have investigated the computational performance of dRTI for large-scale OCPs with up to 333,000 decision variables in simulations with electric power grids. Similarly to [Conte et al. 2012; Behrunani et al. 2024], the number of subsystems in the network was found to have little effect on the required number of optimizer iterations per control step and we observe this phenomenon for a range of problem settings with different grid sizes, load scenarios, and initial disturbances. This promises good scalability since expensive computations are parallelizd between subsystems. In contrast to prior work, we further compared the proposed DMPC approaches to centralized MPC based on parallelized interior point solvers, showing comparatively fast convergence of dRTI to suboptimal control inputs. Moreover, we demonstrated that the resulting suboptimal feedback yields strong control performance thanks to the system-optimizer convergence in closed loop.

7 Summary and Outlook

The increasing prevalence of CPSoS in engineering domains such as manufacturing, robotics, energy systems, and beyond calls for novel optimization and control architectures which scale and leverage modern information technology to deliver high performance. Cooperative DMPC is a promising way forward in this direction as it combines the strengths of centralized MPC and distributed control. However, threats to real-time feasibility can inhibit real-world implementations, because iterative decentralized optimization methods with multiple communication rounds can cause significant delay when solving the OCP to determine the next control input. As such, real-time feasibility is a challenge which is not unique to DMPC, but may also arise in centralized NMPC. There, tailored Real-Time Iteration (RTI) schemes have significantly reduced the computational burden over the last decades, allowing to deploy NMPC at microsecond sampling intervals. It thus stands to reason that RTI schemes can also help accelerate DMPC as long as computations can be efficiently decomposed among subsystems.

7.1 Real-Time Iterations for Distributed MPC

This thesis has developed a decentralized RTI (dRTI) approach for cooperative nonlinear DMPC based on a novel bi-level decentralized Sequential Quadratic Programming (dSQP) method. The dSQP algorithm combines an SQP method with ADMM to solve partially separable NLPs subject to non-convex constraints. Computational advantages are that each subsystem must only solve small-scale convex QPs online and that the linearized inequality constraints are passed to ADMM such that an explicit identification of the active set is avoided. We addressed the following challenges which are crucial for the real-time control of CPSoS.

Stability The early termination in RTI schemes results in OCP solutions which are not only suboptimal, but even infeasible. This complicates the stability analysis for closed-loop control, because standard arguments based on the feasibility of suboptimal solutions are not available. Thus, the stability of the system-optimizer dynamics hinges on the fast local convergence of the employed optimization algorithm, with q-linear convergence to the OCP minimizer being a commonly adopted sufficient condition for closed-loop stability. To this end, we prove that dSQP converges locally at a q-linear rate to regular KKT points if sufficiently many ADMM iterations are applied in each SQP step. The number of ADMM iterations can be kept constant inside the controller and a sufficient number of iterations to guarantee stability can be computed offline. Stability of the system-optimizer dynamics then follows from established guarantees for centralized NMPC under the assumptions that the OCP design is stabilizing and that the control sampling frequency is sufficiently high. The approach is flexible with respect to the adopted OCP design such that different synthesis methods from the DMPC literature, with or without terminal constraints,

can be applied. These stability guarantees improve upon the state of the art as they are, to the best of the author's knowledge, the first stability guarantees for nonlinear DMPC without a centralized coordinator that do not assume—but prove—convergence of the optimization algorithm.

Implementation The primary aim of dRTI is to improve and demonstrate the real-time feasibility of cooperative DMPC in the hope that this helps pave the way towards real-world applications of DMPC in CPSoS. To this end, we have studied the performance of cooperative DMPC based on dRTI in hardware experiments with distributed computation to capture crucial effects such as limited computational resources, imperfect communication, and delays. Six test series on two robotic hardware platforms investigated a wide range of scenarios and allowed to contrast the dRTI performance for convex vs. non-convex OCPs, Ethernet vs. WiFi communication, embedded vs. offboard computation, and static vs. dynamic environments. Moreover, we analyzed the real-time dilemma of MPC by exploring the tradeoff between fast control sampling and more optimizer iterations which, in the context of DMPC, can also be understood as a tradeoff between fast sampling and close collaboration through higher levels of consensus. The experiments demonstrated that the approach indeed is real-time feasible such that the proposed dRTI scheme is, to the best of the author's knowledge, the first DMPC algorithm without centralized coordinator that enjoys theoretical guarantees *and* has been validated in practice. Moreover, the computational efficiency of dSQP allowed to run embedded DMPC at 20 Hz, a fivefold increase over prior results.

Scalability CPSoS are interconnected dynamical systems with many states and inputs and thus the scalability of any DMPC algorithm is vital. Throughout Chapters 2–4, we have considered small- to medium-scale optimization problems such as the IEEE 118-bus AC-OPF example with $n = 576$ decision variables, the swing-up of coupled inverted pendulums with $n = 568$–1504, and hardware experiments with $n = 216$–1504. These problems tackled different challenges such as complicated non-convex constraints, unstable nonlinear dynamics, and real-time collision avoidance, but they did not focus on numerical performance for large problem dimensions. The scalability study in Chapter 6 on the frequency control of electric power grids thus investigated the dRTI performance for convex-quadratic and nonlinear OCPs with $n = 13{,}000$–$333{,}000$ decision variables. Across a wide range of scenarios, we observe that the required number of optimizer iterations needed to converge to a pre-specified accuracy does not depend on the number of subsystems in the CPSoS. This is encouraging with respect to real-time feasibility, because dRTI decomposes computations among subsystems such that the execution time in practice should only depend on the number of optimizer iterations, but not on the number of subsystems. A key improvement over prior scalability studies related to DMPC is that we provide runtimes for state-of-the-art centralized solvers *and* dRTI, all of which we parallelize on a multi-core CPU server. The results demonstrate that centralized interior point solvers with parallelization on multi-core CPUs struggle to produce accurate solutions with moderate solve times. In contrast, dRTI rapidly finds suboptimal OCP solutions and, for convex QPs, is on par with OSQP. Compared to centralized MPC based on in-

terior point methods, dRTI achieves 99 % closed-loop control performance in less than 7 % of the execution time. For the considered example, DMPC thus outperforms centralized MPC from a computational point of view.

7.2 Open Challenges in Distributed MPC: Real-Time Execution and Beyond

This thesis dicussed tailored optimization algorithms for implementing DMPC in real time, an important prerequisite for applying DMPC in practice. Nonetheless, several challenges remain before cooperative DMPC can be expected to deliver on its promise of reconciling centralized performance with distributed control in real-world applications.

Real-time feasibility With respect to numerical optimization and real-time feasibility, the perhaps most fundamental open challenge is the development of optimizaton algorithms which provide a high level of consensus in decentralized implementations. An ideal decentralized optimization algorithm for DMPC would be feasible-side convergent such as Jacobi iterations, but allow for infeasible warm starts like ADMM. However, these specifications contradict one another to some extent and it is not clear how this conflict can be resolved. Future DMPC schemes may thus benefit from structured analyses on the lack of consensus and its implications for closed-loop control to derive design guidelines and recommendations for practitioners on how to cope with asymptotic feasibility.

Asynchronous and interoperable DMPC design and operation Cooperative DMPC as discussed in this thesis essentially follows a top-down approach: First formulate a centralized OCP, then decentralize computations by means of a suitable optimization algorithm. While this methodology does not require subsystems to share individual problem data such as objective functions and system dynamics online, the theoretical stability analysis and offline tuning of the optimization methods utilized the centralized availability of problem information. However, industrial applications may require a bottom-up approach where individual subsystems design their parts of the cooperative OCP autonomously when forming a CPSoS. In a sense, this could be regarded as a design-phase counterpart to asynchronous optimization algorithms which can proceed without receiving all desired information from neighbors. In the context of DMPC, developments into this direction date back at least to the 2010s with the advent of *plug-and-play* DMPC, discussing the dynamic reconfiguration of the centralized OCP if subsystems enter or leave a network. Interoperability, however, requires even further generalizations to the cooperative OCP design and its numerical solution. For instance, is it possible to allow for heterogeneous sampling intervals δ, prediction horizons N, and tuning parameters such as ρ in ADMM across subsystems? Can the DMPC scheme react to changes in the CPSoS without manual retuning, for instance if the control

objectives or system dynamics change? These questions are presently addressed via decentralized MPC and one open challenge for DMPC is to promote interoperability without compromising cooperation and performance.

CPSoS challenges in DMPC Since distributed control schemes form an integral part of the ecosystem in which they are embedded, DMPC inherits some of the challenges generally present in CPSoS. In view of the above, cross-vendor interoperability is an open challenge for DMPC which may necessitate standardization as is commonly adopted in e.g. communication networks. The latter is closely linked to distributed control and in particular to cooperative DMPC, because fast and reliable communication is mandatory for running the controllers. This tight interaction between communication and control has sparked manifold research efforts in both communities such as time sensitive networking to ensure the reliable and timely transmission of highly critical messages or event-triggered control to reduce the load on the communication network. Moreover, joint initiatives are under way to incorporate the requirements of both domains in the respective designs, i.e. by encoding the needs of communication networks directly in the control synthesis and vice versa. A further major challenge in CPSoS are privacy and security, which are in parts already being addressed by tailored DMPC schemes and which can be expected to further impact the design and solution of cooperative OCPs. Finally, applications impose additional technical and economic requirements for DMPC schemes that need to be addressed. For instance, how can subsystems be incentivized to participate in cooperative DMPC even if this will result, at times, in suboptimal performance with respect to the individual objective? These challenges are not unique to DMPC and will require joint efforts between the systems and controls community with other domains to facilitate and promote the control of CPSoS.

A Numerical Optimization Background

This appendix recalls standard concepts from numerical optimization and key convergence results that are used frequently throughout the thesis.

A.1 Nonlinear Programming

Consider the NLP

$$\min_{z} \; f(z) \tag{A.1a}$$

$$\text{subject to} \quad g(z) = 0 \mid \nu, \tag{A.1b}$$

$$h(z) \leq 0 \mid \mu, \tag{A.1c}$$

$$Ez = b \mid \lambda, \tag{A.1d}$$

with the decision variable $z \in \mathbb{R}^n$. We write the linear equality constraint (A.1d) separately instead of incorporating it into (A.1b) in order to harmonize the notation with the other chapters. The functions $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^{n_g}$, and $h : \mathbb{R}^n \to \mathbb{R}^{n_h}$ are assumed to be three times continuously differentiable, $E \in \mathbb{R}^{n_c \times n}$ is assumed to have full row rank, and $b \in \mathbb{R}^{n_c}$. The notation in (A.1) highlights that $\nu \in \mathbb{R}^{n_g}$, $\mu \in \mathbb{R}^{n_h}$, and $\lambda \in \mathbb{R}^{n_c}$ are Lagrange multipliers. The feasible set of NLP (A.1) is given by $\underline{\mathbb{Z}} \doteq \{z \in \mathbb{R}^n \,|\, g(z) = 0, h(z) \leq 0, Ez = b\}$. For $z \in \underline{\mathbb{Z}}$, the sets of active and inactive inequality constraints are defined as $\mathcal{A}(z) \doteq \{j \in \{1, \dots, n_h\} \,|\, [h(z)]_j = 0\}$ and $\mathcal{I}(z) \doteq \{j \in \{1, \dots, n_h\} \,|\, [h(z)]_j < 0\}$, respectively.

Definition A.1 (Local minimizer [Geiger and Kanzow 2002, Definition 1.1])**.** *A point $z^\star \in \underline{\mathbb{Z}}$ is called a local minimizer of NLP* (A.1)*, if there exists an $\varepsilon > 0$ such that $f(z^\star) \leq f(z)$ for all $z \in \underline{\mathbb{Z}} \cap \mathcal{B}(z^\star, \varepsilon)$.* □

Definition A.2 (Strict local minimizer [Geiger and Kanzow 2002, Definition 1.1])**.** *A point $z^\star \in \underline{\mathbb{Z}}$ is called a strict local minimizer of NLP* (A.1)*, if there exists an $\varepsilon > 0$ such that $f(z^\star) < f(z)$ for all $z \in \underline{\mathbb{Z}} \cap \mathcal{B}(z^\star, \varepsilon)$ with $z \neq z^\star$.* □

Definition A.3 (LICQ [Geiger and Kanzow 2002, Definition 2.40])**.** *A point $z \in \underline{\mathbb{Z}}$ satisfies LICQ, if the matrix*

$$\begin{bmatrix} \nabla g(z)^\top \\ [\nabla h(z)]^\top_{\mathcal{A}(z)} \\ E \end{bmatrix}$$

has full row rank, i.e., if the gradients of the equality constraints and the active inequality constraints are linearly independent. □

We define the primal-dual variables $p \doteq (z, \nu, \mu, \lambda) \in \mathbb{R}^{n_p}$ and the Lagrangian to NLP (A.1) as

$$L(z, \nu, \mu, \lambda) \doteq f(z) + \nu^\top g(z) + \mu^\top h(z) + \lambda^\top (Ez - b). \tag{A.2}$$

Definition A.4 (KKT conditions [Geiger and Kanzow 2002, Definition 2.35])**.** *A point* $p^\star = (z^\star, \nu^\star, \mu^\star, \lambda^\star)$ *is called a KKT point of NLP* (A.1)*, if it satisfies the KKT conditions*

$$\nabla_z L(z^\star, \nu^\star, \mu^\star, \lambda^\star) = 0, \tag{A.3a}$$
$$g(z^\star) = 0, \tag{A.3b}$$
$$h(z^\star) \leq 0,\ \mu^\star \geq 0,\ \mu^{\star\top} h(z^\star) = 0, \tag{A.3c}$$
$$Ez^\star = b. \tag{A.3d}$$

□

The *complementarity conditions* (A.3c) are equivalent to $\min(-h(z^\star), \mu^\star) = 0$, where $\min(a, b) \doteq (\min([a]_1, [b]_1), \ldots, \min([a]_n, [b]_n))$ is the vector of componentwise minima for two vectors $a, b \in \mathbb{R}^n$.

Theorem A.1 (Necessary conditions of optimality [Geiger and Kanzow 2002, Theorem 2.41])**.** *Let* $z^\star \in \underline{\mathbb{Z}}$ *be a local minimizer of NLP* (A.1) *which satisfies LICQ. Then there exist unique Lagrange multipliers* $\nu^\star \in \mathbb{R}^{n_g}$*,* $\mu^\star \in \mathbb{R}^{n_h}$*, and* $\lambda^\star \in \mathbb{R}^{n_c}$ *such that* $p^\star = (z^\star, \nu^\star, \mu^\star, \lambda^\star)$ *is a KKT point of NLP* (A.1). □

Definition A.5 (Strict complementarity [Nocedal and Wright 2006, Definition 12.5])**.** *Let* $z^\star$ *be a local minimizer of NLP* (A.1) *and let* $z^\star$ *and* $\mu^\star \in \mathbb{R}^{n_h}$ *satisfy* (A.3c)*. Strict complementarity is said to hold, if exactly one of* $[z^\star]_j$ *and* $[\mu^\star]_j$ *is zero for each index* $j = \{1, \ldots, n_h\}$*, i.e., if* $[\mu^\star]_j > 0$ *for all* $j \in \mathcal{A}(z^\star)$. □

By definition, strict complementarity rules out weakly active inequality constraints at $z^\star$, i.e., there is no constraint j such that $[h(z^\star)]_j = [\mu^\star]_j = 0$. An equivalent definition of strict complementarity is that (A.3c) holds and $\mu^\star + h(z^\star) \neq 0$ [Geiger and Kanzow 2002, p. 47].

Theorem A.2 (SOSC with strict complementarity)**.**
Let $p^\star = (z^\star, \nu^\star, \mu^\star, \lambda^\star)$ *be a KKT point of NLP* (A.1) *and let strict complementarity hold. If*

$$z^\top \nabla^2_{zz} L(z^\star, \nu^\star, \mu^\star) z > 0 \quad \textit{for all} \quad z \neq 0 \textit{ with } (\nabla g(z^\star)^\top, [\nabla h(z^\star)]^\top_{\mathcal{A}(z^\star)}, E) z = 0, \tag{A.4}$$

then $z^\star$ *is a strict local minimizer of NLP* (A.1). □

The conditions (A.4) are called Second-Order Sufficient Conditions (SOSC) and Theorem A.2 is an adapted version of [Nocedal and Wright 2006, Theorem 12.6] for KKT points where strict complementarity holds.

A.2 Sensitivity Analysis

Consider the parametric NLP

$$\min_z f(z,\xi) \quad \text{subject to} \quad g(z,\xi) = 0, \quad h(z,\xi) \leq 0, \tag{A.5a}$$

where $z \in \mathbb{R}^n$ is the vector of decision variables, $\xi \in \mathbb{R}^m$ is a perturbation parameter, $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^{n_g}$, and $h : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^{n_h}$. We refer to the perturbed NLP (A.5) parameterized by ξ as $P(\xi)$. The following Basic Sensitivity Theorem (BST) originally appeared in [Fiacco 1976] and we here recall a version from the textbook [Fiacco 1983, Theorem 3.2.2].

Theorem A.3 (Basic Sensitivity Theorem [Fiacco 1983])**.** *Let $z^\star$ be a local minimizer of $P(0)$. If*

i) the functions f, g, and h in NLP (A.5) *are twice continuously differentiable in z,*

ii) the functions g and h and the Jacobians $\nabla_z f$, $\nabla_z g$, and $\nabla_z h$ are once continuously differentiable in ξ in a neighborhood of $(z^\star, 0)$, and

iii) SOSC, LICQ, and strict complementarity hold at $z^\star$ with Lagrange multipliers $\nu^\star \in \mathbb{R}^{n_g}$ and $\mu^\star \in \mathbb{R}^{n_h}$,

then

a) $z^\star$ is a local isolated minimizer and the Lagrange multipliers $\nu^\star$ and $\mu^\star$ are unique,

b) for ξ in a neighborhood of 0, there exists a unique, once continuously differentiable function $p(\xi) = (z(\xi), \nu(\xi), \mu(\xi))$ satisfying SOSC for $P(\xi)$ and hence $z(\xi)$ is a locally unique local minimizer of $P(\xi)$ with associated Lagrange multipliers $\nu(\xi)$ and $\mu(\xi)$, and

c) for ξ near 0, the set of binding inequalities is unchanged, strict complementarity holds, and LICQ holds at $z(\xi)$. □

We use Theorem A.3 to study the sensitivity of the quadratic subproblems to be solved in dSQP. To this end, we note that part *b*) of the BST can be adapted to obtain, for ξ near 0,

$$\begin{aligned} z^\top \nabla^2_{zz} L(z^\star, \nu^\star, \mu^\star, 0) z > 0 \quad \text{for all} \quad \nabla g(z^\star, 0)^\top z = 0 \\ \implies z^\top \nabla^2_{zz} L(z(\xi), \nu(\xi), \mu(\xi), \xi) z > 0 \quad \text{for all} \quad \nabla g(z(\xi), \xi)^\top z = 0. \end{aligned} \tag{A.6}$$

This follows by adapting the proof in [Fiacco 1983, p. 75] to the stronger SOSC version (A.6).

A.3 Convex Analysis

The design and analysis of optimization algorithms for solving NLP (A.1) depends on the convexity of the NLP components, which we define in this section. To this end, we first recall basic notions of convex analysis from [Nocedal and Wright 2006, Chapter 1] and [Bertsekas 2016, Appendix B].

Definition A.6 (Convex set [Nocedal and Wright 2006])**.**
A set $\mathbb{X} \subseteq \mathbb{R}^n$ is called convex if $\alpha x + (1-\alpha)y \in \mathbb{X}$ for all $x, y \in \mathbb{X}$ and for all $\alpha \in [0,1]$. □

Definition A.7 (Convex and strictly convex function [Nocedal and Wright 2006])**.**
Let $\mathbb{X} \subseteq \mathbb{R}^n$ be a convex set. A function $f : \mathbb{X} \to \mathbb{R}$ is called

1. *convex, if $f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y)$ for all $x, y \in \mathbb{X}$ and for all $\alpha \in [0,1]$,*
2. *strictly convex, if the above inequality is strict for all $x, y \in \mathbb{X}$ with $x \neq y$ and for all $\alpha \in (0,1)$.* □

Definition A.8 (Epigraph, effective domain, and closed and proper function [Bertsekas 2016])**.**
Let $\mathbb{X} \subset \mathbb{R}^n$ be a convex set and let $f : \mathbb{X} \to [-\infty, \infty]$.

1. *The epigraph of f is defined as the set* $\operatorname{epi}(f) \doteq \left\{(x, w) \in \mathbb{R}^{n+1} \;\middle|\; x \in \mathbb{X}, w \in \mathbb{R}, f(x) \leq w\right\}$.
2. *The effective domain of f is defined as the set* $\operatorname{dom}(f) \doteq \{x \in \mathbb{X} \mid f(x) < \infty\}$.
3. *The function f is said to be closed, if* $\operatorname{epi}(f)$ *is a closed set.*
4. *The function f is said to be proper, if* $\operatorname{dom}(f)$ *is nonempty and if $f(x) > -\infty$ for all $x \in \mathbb{X}$.* □

The analysis of Dual Decomposition (DD) assumes the objective functions to be strongly convex. Similar to the definition via [Bertsekas 2016, Equation B.3], we here define strong convexity for continuously differentiable functions.

Definition A.9 (Strongly convex function)**.** *A continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be strongly convex with strong convexity parameter $\mu_c > 0$ if*

$$f(y) \geq f(x) + \nabla f(x)^\top (y - x) + \frac{\mu_c}{2} \|x - y\|^2 \quad \forall x, y \in \mathbb{R}^n. \tag{A.7}$$

□

The above definition is a slight modification to the definition in [Bertsekas 2016] as we additionally define the strong convexity parameter μ_c. This is motivated by the analysis of dual decomposition presented in [Braun and Grüne 2018] and we refer to [Braun 2016] for a more detailed discussion of strong convexity.

Theorem A.4 (Strong convexity [Bertsekas 2016, Proposition B.5 (b)])**.** *If $f : \mathbb{R}^n \to \mathbb{R}$ is a twice continuously differentiable function, then f satisfies* (A.7) *if and only if the matrix $\nabla^2 f(x) - \mu_c I$ is positive semi-definite for all $x \in \mathbb{R}^n$.* □

Theorem A.4 serves as a practical guide for checking strong convexity in implementations. That is, a twice continuously differentiable function f is strongly convex with convexity parameter μ_c if and only if the smallest eigenvalue of the Hessian satisfies $\lambda_{\min}(\nabla^2 f(x)) \geq \mu_c$ for all $x \in \mathbb{R}^n$.

A.4 SQP Convergence

SQP methods are centralized algorithms for solving NLP (A.1) which proceed by solving a sequence of quadratic subproblems [Boggs and Tolle 1995; Nocedal and Wright 2006]. This section presents the centralized QP to be solved in each SQP step, discusses the uniqueness of the QP solution, and summarizes a classic SQP convergence result for regular KKT points.

Consider a QP approximation of NLP (A.1) at a primal-dual point $p^k \doteq (z^k, \nu^k, \mu^k, \lambda^k) \in \mathbb{R}^{n_p}$,

$$\min_z \frac{1}{2}(z - z^k)^\top H^k (z - z^k) + \nabla f^{k\top}(z - z^k) \tag{A.8a}$$

$$\text{subject to} \quad g^k + \nabla g^{k\top}(z - z^k) = 0 \mid \nu, \tag{A.8b}$$

$$h^k + \nabla h^{k\top}(z - z^k) \leq 0 \mid \mu, \tag{A.8c}$$

$$Ez = b \mid \lambda, \tag{A.8d}$$

where $H^k \approx \nabla^2_{zz} L(z^k, \nu^k, \mu^k)$ is symmetric. The symbols g^k and ∇g^k are shorthands for $g(z^k)$ and $\nabla g(z^k)$, respectively and the same holds for functions f and h. We denote a primal dual-solution to QP (A.8) as $p^{k,\star} = (z^{k,\star}, \nu^{k,\star}, \mu^{k,\star}, \lambda^{k,\star})$ and recall the following standard SQP convergence result.

Definition A.10 (Convergence rates [Ulbrich and Ulbrich 2012, Definition 10.2])**.** *The sequence* $\{p^k\} \subset \mathbb{R}^{n_p}$ *is said to converge to* $p^\star \in \mathbb{R}^{n_p}$

i) q-linearly, if $\|p^{k+1} - p^\star\| \leq c\|p^k - p^\star\|$ *for all* $k \geq k_0$ *and some* $c < 1$ *and* $k_0 \geq 0$*,*

ii) q-superlinearly, if $p^k \to p^\star$ *and* $\|p^{k+1} - p^\star\| = o(\|p^k - p^\star\|)$ *for* $k \to \infty$*,*

iii) q-quadratically, if $p^k \to p^\star$ *and* $\|p^{k+1} - p^\star\| = O(\|p^k - p^\star\|^2)$ *for* $k \to \infty$. □

An equivalent definition for q-quadratic convergence is to say there exists a constant $C > 0$ such that $\|p^{k+1} - p^\star\| \leq C\|p^k - p^\star\|^2$ for all $k \geq 0$ [Ulbrich and Ulbrich 2012, Definition 10.2].

Theorem A.5 (Centralized SQP convergence)**.** *Let* $p^\star = (z^\star, \nu^\star, \mu^\star, \lambda^\star)$ *denote a KKT point of NLP* (A.1) *which satisfies strict complementarity, LICQ, and SOSC* (A.4)*. Consider an exact-Hessian SQP scheme with* $H^k = \nabla^2_{zz} L(z^k, \nu^k, \mu^k)$*, where the next iterate* p^{k+1} *is set to the KKT point* $p^{k,\star}$ *of QP* (A.8) *closest to* p^k*. Then, there exists a constant* $\varepsilon_0 > 0$ *such that the sequence* $\{p^k\}$ *generated by the centralized SQP scheme converges q-quadratically to* $p^\star$ *for all* $p^0 \in \mathcal{B}(p^\star, \varepsilon_0)$. □

Proof. Recall that we assume the maps f, g, and h in NLP (A.1) to be three times continuously differentiable. Therefore, the Hessians $\nabla^2 f$, $\nabla^2[g]_i$, $i \in \{1, \ldots, n_g\}$ and $\nabla^2[h]_j$, $j \in \{1, \ldots, n_h\}$ are Lipschitz continuous if $p^k \approx p^\star$. Local q-quadratic convergence therefore follows from [Geiger and Kanzow 2002, Theorem 5.31]. ■

We define the map

$$F(p) \doteq \begin{bmatrix} \nabla_z L(z, \nu, \mu, \lambda) \\ g(z) \\ \min(-h(z), \mu) \\ Ez - c \end{bmatrix}.$$

The convergence proof presented in [Geiger and Kanzow 2002] shows that, inside $\mathcal{B}(p^\star, \varepsilon_0)$, the centralized SQP iterates $\{p^k\}$ are equivalent to the iterates generated by Newton's method applied to the nonlinear system of equations $F(p) = 0$. Hence, the SQP scheme inherits the fast local convergence from Newton's method. In particular, for all $p \in \mathcal{B}(p^\star, \varepsilon_0)$, $F(p)$ is continuously differentiable, the Jacobian $\nabla F^{k\top} \doteq \nabla F(p^k)^\top$ is regular, and the Newton iteration

$$F^k + \nabla F^{k\top}(p^{k+1} - p^k) = 0$$

is well defined, where $F^k \doteq F(p^k)$.

Furthermore, observe that QP (A.8) may have multiple KKT points under the assumptions in Theorem A.5, even if $p^k \approx p^\star$ [Geiger and Kanzow 2002]. This can be addressed in the theoretical analysis of SQP methods by defining the next iterate p^{k+1} of the SQP scheme to be the KKT point of QP (A.8) which is closest to p^k [Geiger and Kanzow 2002, Algorithm 5.30], cf. [Robinson 1974]. Here, we instead strengthen the SOSC assumption on $p^\star$ to guarantee that QP (A.8) indeed only has one KKT point. We define

$$\mathbb{Z}^k \doteq \left\{ z \in \mathbb{R}^n \;\middle|\; \begin{matrix} g^k + \nabla g^{k\top}(z - z^k) = 0 \\ h^k + \nabla h^{k\top}(z - z^k) \leq 0 \end{matrix} \right\}.$$

Lemma A.1 (Unique solution of the SQP subproblem)**.** *Let QP* (A.8) *be feasible, let* $\nabla g^{k\top}$ *have full row rank, and let*

$$z^\top H^k z > 0 \quad \textit{for all} \quad z \neq 0 \quad \textit{with} \quad \nabla g^{k\top} z = 0. \tag{A.9}$$

Then, the objective function

$$f^{\mathrm{QP},k}(z) \doteq \frac{1}{2}(z - z^k)^\top H^k (z - z^k) + \nabla f^{k\top}(z - z^k)$$

is strictly convex over $\mathbb{Z}^k$*, that is,*

$$f^{\mathrm{QP},k}\left((1-\lambda)x + \lambda y\right) < (1-\lambda) f^{\mathrm{QP},k}(x) + \lambda f^{\mathrm{QP},k}(y)$$

for all $x, y \in \mathbb{Z}^k$ *with* $x \neq y$ *and all* $\lambda \in (0, 1)$*. Furthermore, there exists a unique global minimizer* $z^{k,\star}$ *of QP* (A.8)*.*

If $z^{k,\star}$ *additionally satisfies LICQ, i.e., if the matrix*

$$\begin{bmatrix} \nabla g^{k\top} \\ [\nabla h^k]^\top_{\mathcal{A}(z^{k,\star})} \\ E \end{bmatrix} \tag{A.10}$$

has full row rank, then there exists a unique KKT point $p^{k,\star} \doteq (z^{k,\star}, \nu^{k,\star}, \mu^{k,\star}, \lambda^{k,\star})$ *of QP* (A.8)*.* □

Proof. The proof proceeds in four steps. First, a), we show that the reduced Hessian is positive definite on the null space of the equality constraints. Then, b), we show that f^{QP} is strictly convex

over $\mathbb{Z}^k$. Then, c), we show that a strict global minimum $z^{k,\star}$ exists and that there is no other local minimum. Then, d), LICQ implies uniqueness of the Lagrange multipliers and thus of the KKT point $p^{k,\star}$.

a) For any matrix $A \in \mathbb{R}^{m\times n}$, the fundamental theorem of linear algebra states that $\mathbb{R}^n = \text{Null}(A) \oplus \text{Range}(A^\top)$, where the null and range spaces are defined as [Nocedal and Wright 2006, p. 603]

$$\text{Null}(A) \doteq \{z \in \mathbb{R}^n \mid Az = 0\} \quad \text{and} \quad \text{Range}(A^\top) \doteq \{z \in \mathbb{R}^n \mid z = A^\top w \text{ for some } w \in \mathbb{R}^m\}.$$

Because $\nabla g^{k\top} \in \mathbb{R}^{n_g\times n}$ has full row rank, we can decompose any vector $(z - z^k) \in \mathbb{R}^n$ via the null space method [Nocedal and Wright 2006, Chapter 16] as

$$z - z^k \doteq Zv + \nabla g^k w, \tag{A.11}$$

into components $v \in \mathbb{R}^{n-n_g}$ and $w \in \mathbb{R}^{n_g}$. The matrix $Z \in \mathbb{R}^{n\times(n-n_g)}$ is a null space basis of $\nabla g^{k\top}$, i.e., $\nabla g^{k\top} Z = 0$. We can rewrite the stronger version of SOSC (A.9) as

$$(z - z^k)^\top H^k (z - z^k) > 0 \quad \text{for all} \quad z \neq z^k \quad \text{with} \quad \nabla g^{k\top}(z - z^k) = 0. \tag{A.12}$$

Inserting (A.11) into (A.12) yields

$$v^\top Z^\top H^k Z v > 0 \quad \text{for all} \quad v \in \mathbb{R}^{n_g}, \tag{A.13}$$

where we have used that the matrix $\nabla g^{k\top}\nabla g^k$ is invertible and thus $\nabla g^{k\top}(z - z^k) = 0$ implies $w = 0$. Hence, the reduced Hessian $\bar{H} = Z^\top H^k Z$ is positive definite.

b) Let $z \in \mathbb{Z}^k$ and thus

$$g^k + \nabla g^{k\top}(z - z^k) = 0. \tag{A.14}$$

Inserting the null space approach (A.11) into the constraint (A.14) yields $w = -(\nabla g^{k\top}\nabla g^k)^{-1} g^k$, i.e., w is unique. Hence, for any points $x, y \in \mathbb{Z}^k$, we can write the difference $(x - z^k) - (y - z^k)$ as

$$x - z^k - (y - z^k) \doteq Zv_x + \nabla g^k w - (Zv_y + \nabla g^k w) = Z(v_x - v_y) \tag{A.15}$$

for some $v_x, v_y \in \mathbb{R}^{n-n_g}$. Note that $f^{\text{QP},k}$ is quadratic and thus continuously differentiable everywhere. Hence, $f^{\text{QP},k}$ is strictly convex over $\mathbb{Z}^k$ if and only if [Ulbrich and Ulbrich 2012, Theorem 6.3]

$$\nabla f^{\text{QP},k}(x)^\top (y - x) < f^{\text{QP},k}(y) - f^{\text{QP},k}(x) \tag{A.16}$$

for all $x, y \in \mathbb{Z}^k$ and $x \neq y$. We next show that (A.16) indeed holds. Define the shorthands $\Delta x \doteq x - z^k$ and $\Delta y \doteq y - z^k$ and note that $y - x = \Delta y - \Delta x$. Inserting the definition of $f^{\text{QP},k}$ into (A.16) yields, for all $x, y \in \mathbb{Z}^k$ with $x \neq y$,

$$\begin{aligned}(H^k \Delta x + \nabla f^k)^\top (y - x) &< \frac{1}{2}\Delta y^\top H^k \Delta y - \frac{1}{2}\Delta x^\top H^k \Delta x + \nabla f^{k\top}(y - x)\\ (H^k \Delta x)^\top (y - x) &< \frac{1}{2}\Delta y^\top H^k \Delta y - \frac{1}{2}\Delta x^\top H^k \Delta x\end{aligned}$$

$$
\begin{aligned}
(H^k \Delta x)^\top (\Delta y - \Delta x) &< \frac{1}{2}\Delta y^\top H^k \Delta y - \frac{1}{2}\Delta x^\top H^k \Delta x \\
0 &< \frac{1}{2}\Delta x^\top H^k \Delta x - \Delta x^\top H^k \Delta y + \frac{1}{2}\Delta y^\top H^k \Delta y \\
0 &< \frac{1}{2}(\Delta x - \Delta y)^\top H^k (\Delta x - \Delta y) \\
0 &< \frac{1}{2}(x - y)^\top H^k (x - y).
\end{aligned}
$$

Inserting (A.15) into the last inequality yields, for all $v_x, v_y \in \mathbb{R}^{n-n_g}$ with $v_x \neq v_y$,

$$
\begin{aligned}
0 &< \frac{1}{2}(Z(v_x - v_y))^\top H^k (Z(v_x - v_y)) \\
0 &< \frac{1}{2}(v_x - v_y)^\top Z^\top H^k Z (v_x - v_y).
\end{aligned}
$$

The last inequality holds, because the reduced Hessian $\bar{H}^k = Z^\top H^k Z$ is positive definite. Thus, $f^{\mathrm{QP},k}$ is indeed strictly convex over $\mathbb{Z}^k$.

c) QP (A.8) is feasible by the assumptions stated in the theorem. Furthermore, the objective $f^{\mathrm{QP},k}$ is bounded below over $\mathbb{Z}^k$, because $\bar{H}^k$ is positive definite. Thus,

$$
f_{\mathbb{Z}^k}^{\mathrm{QP},k,\star} \doteq \inf_{z \in \mathbb{Z}^k} f^{\mathrm{QP},k}(z) \quad = \quad \begin{aligned} &\inf_{v \in \mathbb{R}^{n-n_g}} f^{\mathrm{QP},k}(Zv + \nabla g^k w) \\ &\text{s.t.} \quad h^k + \nabla h^{k\top}(Zv + \nabla g^k w) \leq 0 \\ &\qquad\; w = -(\nabla g^{k\top} \nabla g^k)^{-1} g^k \end{aligned} \quad > \quad -\infty,
$$

i.e., the *optimal value* $f_{\mathbb{Z}^k}^{\mathrm{QP},k,\star}$ is finite. The Frank-Wolfe theorem [Frank and Wolfe 1956] thus asserts that there exists a solution for the QP [Bertsekas 2016, Exercise 3.1.19]

$$
\min_{z \in \mathbb{Z}^k} f^{\mathrm{QP},k}(z). \tag{A.17}
$$

At the same time, because $f^{\mathrm{QP},k}$ is strictly convex over $\mathbb{Z}^k$, problem (A.17) has at most one minimum which is a strict global minimum [Ulbrich and Ulbrich 2012, Theorem 6.5]. Because the feasible set $\underline{\mathbb{Z}}^k \doteq \mathbb{Z}^k \cap \{z \in \mathbb{R}^n \mid Ez = c\}$ of QP (A.8) is a subset of $\mathbb{Z}^k$, $f^{\mathrm{QP},k}$ is bounded below and strictly convex on $\underline{\mathbb{Z}}^k$. Thus, by the same arguments as for QP (A.17), QP (A.8) has a unique global minimizer $z^{k,\star}$ and no further local minimizers.

d) Because LICQ holds at $z^{k,\star}$, Theorem A.1 guarantees the existence of unique Lagrange multipliers $\nu^{k,\star}$, $\mu^{k,\star}$, and $\lambda^{k\star}$ such that $p^{k,\star} = (z^{k,\star}, \nu^{k,\star}, \mu^{k,\star}, \lambda^{k,\star})$ is a KKT point of QP (A.8). Moreover, $p^{k,\star}$ is the only KKT point, because $z^{k,\star}$ is the only minimizer. ■

Theorem A.6 (Centralized SQP convergence with unique $p^{k,\star}$)**.** *Let $p^\star = (z^\star, \nu^\star, \mu^\star, \lambda^\star)$ denote a KKT point of NLP* (A.1) *which satisfies strict complementarity, LICQ, and*

$$
z^\top \nabla^2_{zz} L(z^\star, \nu^\star, \mu^\star) z > 0 \quad \textit{for all} \quad \nabla g(z^\star)^\top z = 0. \tag{A.18}
$$

Then, there exists a constant $\varepsilon_1 > 0$ such that the following holds. If $p^0 \in \mathcal{B}(p^\star, \varepsilon_1)$, then QP (A.8) *has a unique KKT point $p^{k,\star}$. Moreover, the iterates $\{p^k\}$ produced by a centralized exact-Hessian SQP scheme with $H^k = \nabla^2_{zz} L(z^k, \nu^k, \mu^k)$ and update $p^{k+1} = p^{k,\star}$ converge q-quadratically to $p^\star$.* □

Proof. By the assumptions stated in the theorem, the solution $p^{k,\star}$ to QP (A.8) formed at $p^\star$ satisfies LICQ, the stronger SOSC (A.18), and strict complementarity with the same active inequality constraints as $p^\star$. We can regard QP (A.8) formed at p^k as a perturbed version of the QP formed at $p^\star$ by setting $p^k = p^\star + \xi^k$, where ξ is the perturbation parameter in the BST. Because f, g, and h are three time continuously differentiable, the components of H^k, ∇f^k, ∇g^k, and ∇h^k of QP (3.2) are continuously differentiable with respect to ξ, as required by the BST. The BST thus asserts that there exists a constant $\varepsilon_1 > 0$ such that, for all $p^k \in \mathcal{B}(p^\star, \varepsilon_1)$, the active set stays constant and strict complementarity as well as LICQ also hold for the solution to the perturbed QP (A.8). The stronger SOSC condition

$$y^\top \nabla^2_{zz} L(z^k, \nu^k, \mu^k) y > 0 \quad \text{for all} \quad y \neq 0 \quad \text{with} \quad \nabla g^{k\top} y = 0 \tag{A.19}$$

also carries over to the perturbed QP, which follows from similar arguments as in [Fiacco 1983, p. 75]. Thus, Lemma A.1 guarantees the uniqueness of the KKT point $p^{k,\star}$ and we can drop the specification of Theorem A.5 to take the KKT point closest to p^k. By further restricting ε_1 to be smaller than ε_0 from Theorem A.5, we obtain q-quadratic convergence. ∎

A.5 Linear Convergence of ADMM for Convex QPs

There exists a wide range of ADMM variants and a plethora of options to reformulate a given problem such that ADMM becomes applicable. This section recalls a linear convergence result for the specific ADMM variant and two-block QP formulation considered in this thesis. Such QPs arise in the context of linear-quadratic DMPC or as subproblem QPs in dSQP, for instance.

Consider the feasible two-block QP

$$\min_{y,z} \quad \frac{1}{2}(y-c)^\top H (y-c) + f^\top (y-c) \tag{A.20a}$$

$$\text{subject to} \quad g + J_g(y-c) = 0 \mid \nu, \tag{A.20b}$$

$$h + J_h(y-c) \leq 0 \mid \mu, \tag{A.20c}$$

$$Ez = b \mid \lambda, \tag{A.20d}$$

$$y - z = 0 \mid \gamma, \tag{A.20e}$$

where $H \in \mathbb{R}^{n\times n}$ is symmetric, $y, z, c, \gamma, f \in \mathbb{R}^n$, $g \in \mathbb{R}^{n_g}$, $J_g \in \mathbb{R}^{n_g \times n}$ has full row rank, $h \in \mathbb{R}^{n_h}$, $J_h \in \mathbb{R}^{n_h \times n}$, $E \in \mathbb{R}^{n_c \times n}$ has full row rank, and $b \in \mathbb{R}^{n_c}$. QP (A.20) is said to be in two-block form, because it contains two blocks of decision variables, y and z. We define the polyhedral constraint sets

$$\mathbb{Z} \doteq \left\{ y \in \mathbb{R}^n \;\middle|\; \begin{matrix} g + J_g(y-c) = 0 \\ h + J_h(y-c) \leq 0 \end{matrix} \right\} \quad \text{and} \quad \mathbb{E} \doteq \{ z \in \mathbb{R}^n \mid Ez = b \},$$

and the augmented Lagrangian

$$L_\rho(y, z) \doteq \frac{1}{2}(y-c)^\top H (y-c) + f^\top (y-c) + \gamma^\top (y-z) + \frac{\rho}{2} \|y-z\|_2^2,$$

where $\rho > 0$ is a penalty parameter. The ADMM updates at iteration $l \in \mathbb{N}_0$ read

$$y^{l+1} = \operatorname*{argmin}_{y \in \mathbb{Z}} L_\rho(y, z^l, \gamma^l) \tag{A.21a}$$

$$z^{l+1} = \operatorname*{argmin}_{z \in \mathbb{E}} L_\rho(y^{l+1}, z, \gamma^l) \tag{A.21b}$$

$$\gamma^{l+1} = \gamma^l + \rho(y^{l+1} - z^{l+1}). \tag{A.21c}$$

Theorem A.7 (Linear convergence of ADMM for strictly convex QPs)**.** *Let the Hessian H in QP* (A.20) *be positive definite and denote the unique minimizer of QP* (A.20) *by $y^\star = z^\star$. Let LICQ hold at $y^\star$, i.e., let the matrix $(J_g, [J_h]_{\mathcal{A}(y^\star)}, E)$ have full row rank, and denote the unique consensus Lagrange multipler at the KKT point by $\gamma^\star$. Define the vector $w \doteq (z, \gamma/\rho)$. Then, there exists a constant $a_1 < 1$ such that, for any $w^0 \in \mathbb{R}^{2n}$, the iterates $\{w^l\}$ produced by ADMM satisfy*

$$\|w^{l+1} - w^\star\|_2 \leq a_1 \|w^l - w^\star\|_2 \quad \forall l \in \mathbb{N}_0. \tag{A.22}$$

□

Proof. The presented convergence result is a special case of [Yang and Han 2016, Theorem 14] and we obtain (A.22) by exploiting the fact that $z^\star$ is unique because H is positive definite and by setting $A = I$, $B = -I$, and $b = 0$, where A, B, and b are in the notation of [Yang and Han 2016]. ■

The results in [Yang and Han 2016] extend to more general convex problems, including problems with positive semi-definite Hessian, piecewise linear-quadratic objective functions, and more general two-block consensus constraints. An overview of further linear convergence results for ADMM is given in [Han 2022].

Theorem A.7 guarantees linear convergence if the Hessian H in QP (A.20) is positive definite. However, in the context of SQP methods, $H^k \doteq \nabla^2_{zz} L(z^k, \nu^k, \mu^k)$ is not necessarily positive definite, even close to a regular KKT point. Instead, H^k is only guaranteed to be positive definite on the null space of the constraints. ADMM converges nonetheless, as we show next.

Theorem A.8 (Linear convergence of ADMM for the SQP subproblem)**.** *Let the Hessian H in QP* (A.20) *be positive definite on the null space of the equality constraints* (A.20b)*, i.e., let*

$$z^\top H z > 0 \quad \textit{for all} \quad z \neq 0 \quad \textit{with} \quad J_g z = 0. \tag{A.23}$$

Then, QP (A.20) *has a unique minimizer $y^\star = z^\star$. Furthermore, let LICQ hold at $y^\star$, i.e., let the matrix $(J_g, [J_h]_{\mathcal{A}(y^\star)}, E)$ have full row rank, and denote the unique consensus Lagrange multipler at the KKT point by $\gamma^\star$. Consider the vector $w = (z, \gamma/\rho)$. Then, there exists a constant $a_1 < 1$ such that, for any $w^0 \in \mathbb{R}^{2n}$, the iterates $\{w^l\}$ produced by ADMM satisfy*

$$\|w^{l+1} - w^\star\|_2 \leq a_1 \|w^l - w^\star\|_2 \quad \forall l \in \mathbb{N}_0. \tag{A.24}$$

□

Proof. The proof proceeds in two steps. First, a), we show that the minimizer $y^\star$ is indeed unique. Then, b), we use the strict convexity of the objective function over the constraint set to invoke a standard ADMM convergence result.

a) QP (A.20) can be regarded as a reformulation of the SQP subproblem (A.8) into two-block form. By assumption (A.23), QP (A.8) satisfies the conditions of Lemma A.1 and thus has a unique global minimizer and a unique KKT point. The reformulation into two-block form preserves uniqueness of the solution, which we show by contradiction. Suppose the two-block QP (A.20) has a local minimizer $\tilde{y}^\star = \tilde{z}^\star \neq y^\star = z^\star$. Then, $\tilde{z}^\star$ is also a local minimizer of the single-block QP (A.8). However, by Lemma A.1, QP (A.8) only has a single minimizer $z^\star$, which is a contradition. Thus, QP (A.20) has a unique global minimizer. Moreover, the two-block QP (A.20) also satisfies LICQ and thus the KKT point is unique.

b) The ADMM convergence guarantee in [Yang and Han 2016] is stated for a convex objective function $f : \mathbb{R}^n \to \mathbb{R}$ to be minimized over a polyhedron $\mathcal{X} \subseteq \mathbb{R}^n$, where $x \in \mathbb{R}^n$, f, and $\mathcal{X}$ refer to the first block in ADMM and are in the notation of [Yang and Han 2016]. Crucially, the analysis in [Yang and Han 2016] also applies if f only is convex over the constraint set $\mathcal{X}$. Because of Lemma A.1, the objective (A.20a) is strictly convex over the constraint set $\mathbb{Z}$. Thus, we can invoke Theorem [Yang and Han 2016, Theorem 14] and obtain the q-linear ADMM convergence (A.24). ∎

We note that the analysis in [Yang and Han 2016] and hence Theorems A.7 and A.8 also hold in the absence of the constraints (A.20b)–(A.20d), even though $\mathbb{R}^n$ strictly speaking is not a polyhedron.

A.6 Essentially Decentralized Conjugate Gradients

We consider the linear system of equations

$$\Big(\underbrace{\sum_{i\in\mathcal{S}} S_i}_{\doteq S}\Big)\lambda = \underbrace{\sum_{i\in\mathcal{S}} s_i}_{\doteq s}, \tag{A.25}$$

where $\lambda, s \in \mathbb{R}^{n_c}$, $S_i = S_i^\top \in \mathbb{R}^{n_c \times n_c}$ for all $i \in \mathcal{S}$, and S is symmetric positive definite. This type of problem appears for instance in each outer iteration of BL-ALADIN, d-ASM, and d-IP [Engelmann et al. 2020; Stomberg et al. 2021; Engelmann et al. 2021b]. Therein, the matrices S_i inherit favorable sparsity properties from the coupling matrices E_i of the partially separable program (2.1). The sparsity allows to decentralize the well-known conjugate gradient method [Nocedal and Wright 2006].

This section briefly recalls the d-CG algorithm. Further details, including an extension to the case where S is only symmetric positive semi-definite, can be found in [Engelmann and Faulwasser 2021].

Algorithm A.1 d-CG [Engelmann 2020]

1: Initialization: $l = 0$, λ^0, $r^0 = p^0 = s - S\lambda^0$, ε
2: $\lambda_i^0 = I_{C(i)}\lambda^0 =$, $r_i^0 = I_{C(i)}r^0$, $p_i^0 = I_{C(i)}p^0$, and $\eta_i^0 = r_i^{0\top}\Lambda_i^{-1}r_i^0$ for all $i \in \mathcal{S}$
3: $\eta^0 = \sum_{i\in\mathcal{S}} \eta_i^0$
4: **while** $\|r_i^l\|_\infty > \varepsilon$ for all $i \in \mathcal{S}$ **do**
5: $\quad \sigma_i^l = p_i^{l\top}\hat{S}_i p_i^l$
6: $\quad \sigma^l = \sum_{i\in\mathcal{S}} \sigma_i^l$
7: $\quad \lambda_i^{l+1} = \lambda_i^l + \frac{\eta^l}{\sigma^l} p_i^l$
8: $\quad r_i^{l+1} = r_i^l - \frac{\eta^l}{\sigma^l} \sum_{j\in\mathcal{N}_i\cup i} I_{ij}\hat{S}_j p_j^l$
9: $\quad \eta_i^{l+1} = r_i^{l+1\top}\Lambda_i^{-1}r_i^{l+1}$
10: $\quad \eta^{l+1} = \sum_{i\in\mathcal{S}} \eta_i^{l+1}$
11: $\quad p_i^{l+1} = r_i^{l+1} + \frac{\eta^{l+1}}{\eta^l} p_i^l$
12: $\quad l \leftarrow l + 1$
13: **end while**

For all $i \in \mathcal{S}$, we define the set of constraints assigned to subsystem i as

$$C(i) \doteq \left\{ j \in \{1, \dots, n_c\} \,\middle|\, [E_i]_j \neq 0 \right\}.$$

We map from centralized to subsystem variables using the matrix

$$I_{C(i)} \doteq \left(e_j^\top\right)_{j\in C(i)} \in \mathbb{R}^{n_{c_i}\times n_c},$$

where $e_j \in \mathbb{R}^{n_c}$ is the j-th unit vector and $n_{c_i} \doteq |C(i)|$. For this section, we define the neighborhood of subsystem i as $\mathcal{N}_i \doteq \{j \in \mathcal{S} \setminus \{i\} \mid I_C(i)I_{C(j)^\top} \neq 0\}$. This definition is equivalent to (1.4), if (A.25) is derived from a partially separable program in BL-ALADIN, d-ASM, and d-IP. The alternative definition presented here also allows to apply d-CG (A.25) to more general problems than as inner algorithm of the before-mentioned bi-level algorithms.

Algorithm A.1 summarizes d-CG, where $\Lambda \doteq \sum_{i\in\mathcal{S}} I_{C(i)}^\top I_{C(i)}$, $\Lambda_i \doteq I_{C(i)}\Lambda I_{C(i)}^\top$, $\hat{S}_i \doteq I_{C(i)}S_i I_{C(i)}^\top$, and $I_{ij} \doteq I_{C(i)}I_{C(j)}^\top$. The d-CG iterates for each subsystem are the coupling Lagrange multipliers $\lambda_i \in \mathbb{R}^{n_{c_i}}$, the residual $r_i \in \mathbb{R}^{n_{c_i}}$, the step direction $p_i \in \mathbb{R}^{n_{c_i}}$, and the step length $\sigma/\eta \in \mathbb{R}$. The residual update (8) requires the communication of $2n_c$ floats across for the entire network on a neighbor-to-neighbor basis, i.e., a similar communication demand as one iteration of DD or ADMM. Additionally, Steps 3, 6, and 10 sum the scalars σ_i and η_i over all subsystems. These summations can be carried out via hopping and without a central coordinator and we therefore refer to d-CG as an essentially decentralized algorithm [Engelmann 2020, Remark 12].

B Implementational Details for Hardware Experiments

B.1 Mobile Robots

We have implemented two versions of Algorithm 5.1 in C++ and the source code is available online.[1][2] The first version considers linear-quadratic DMPC using ADMM and the second version implements nonlinear DMPC using dSQP. In addition to the C++ standard library, we rely on four external libraries: (i) Eigen[3] to store vectors and matrices; (ii) qpOASES [Ferreau et al. 2014] to solve the subsystem QPs in Step 10; (iii) CasADi [Andersson et al. 2019] to evaluate derivatives in Step 7 using automatic differentiation; and (iv) LCM to communicate states and inputs in Step 3 as well as predicted state trajectories in Steps 11 and 13. To reduce the online execution time, we allocate any dynamic memory and initialize qpOASES prior to running the controllers. Moreover, we use the efficient hotstart procedure of qpOASES to update the current position and velocity in the OCP.

The communication of position measurements, control inputs, and optimizer iterates occurs via the LCM library and follows a publish-subscribe pattern. Each subsystem runs one DMPC program which maintains a separate thread for receiving and caching incoming messages. Algorithm 5.1 fetches data from the cache and waits for all required data before proceeding with the next step such that the subsystems stay synchronized. Since the employed User Datagram Protocol (UDP) is only a best-effort protocol, we specify a maximum waiting time for the position measurements so that the algorithm may proceed if a state message is lost. For the optimizer steps of dSQP and ADMM, however, subsystems always wait until all required messages have arrived to maintain synchrony. To this end, we first synchronize all subsystems at the startup of the programs by exchanging messages with neighbors. During this synchronization step, we check and retransmit missed messages on the application layer to account for heterogeneous launch times of the individual programs. For the ensuing optimizer execution in closed-loop control, however, subsystems do not retransmit potentially lost messages. While we have not observed any difficulties in practice, we note that this approach may in principle result in deadlocks if messages are lost, because subsystems would wait indefinitely for neighbors. To prevent difficulties originating from packet loss, an alternative would be to use the transmission control protocol instead of UDP when implementing ADMM as done in [Burk et al. 2021a].

Remark B.1 (Relation to [Bhanderi et al. 2021] and optimization over 5G networks)**.** *Prior to the above ADMM implementation used in the robot experiments, a separate ADMM code was created*

[1] `https://github.com/OptCon/dmpc_rto`

[2] We gratefully acknowledge openly available code by Carlo Cappello and Petr Listov which we used in our DMPC implementation for reading matrices from disk and for converting between Eigen and CasADi, respectively.

[3] `https://eigen.tuxfamily.org`

at TU Dortmund University for execution on 5G communication networks [Bhanderi et al. 2021]. The implementations of [Bhanderi et al. 2021] and [Stomberg et al. 2023] are written in C++ and they rely on the same third-party libraries. As done in [Bhanderi et al. 2021], the implementation in [Stomberg et al. 2023] stores Eigen matrices in row major format and we follow the approach of [Bhanderi et al. 2021] for passing Eigen variables to qpOASES. Furthermore, parts of the code of [Bhanderi et al. 2021] were subsequently reused in the implementation of [Stomberg et al. 2023], namely for specifying the print options of qpOASES and for storing the QP solutions found by qpOASES in Eigen variables. Apart from these similarities, the two implementations adopt different software architectures and implement different ADMM variants as [Bhanderi et al. 2021] implements the ADMM version presented in [Houska et al. 2016], cf. [Stomberg et al. 2021, Algorithm 3]. We further note that [Bhanderi et al. 2021] not only discusses ADMM, but also presents C++ implementations of d-ASM and d-CG which, together with ADMM and dSQP implementations similar to the ones of Chapter 5, were tested in combination with time-sensitive networking in [Kurtz et al. 2022]. There, DMPC served as an application example to showcase the potential of advanced communication networks for future CPSoS, demonstrating reliable packet transmission for the highly critical DMPC algorithms in the presence of low-criticality cross traffic. □

B.2 Hovercraft

Similar to the code for the robot experiments, the C++ implementation of Algorithm 5.2 and further modules required for controlling the hovercraft is available online.[4] The dRTI code is derived from the code used in the mobile robots experiments and uses Eigen for storing vectors and matrices and CasADi for evaluating derivatives. Instead of the LCM library, ROS2 is used for communication to integrate dRTI into the hovercraft software framework. The subsystem QPs in Step 10 are solved via the sparse interface of the Proximal Interior-Point Quadratic Programming (PIQP) solver [Schwan et al. 2023]. As mentioned in Chapter 5, the subsystems do not wait indefinitely for neighboring messages and proceed asynchronously if necessary. Specifically, in each of Steps 11 and 13 of Algorithm 5.2, each subsystem waits at most 25 ms or until 75 % of the control sampling interval have elapsed, whichever is earlier. We first initialize Algorithm 5.2 with $u_i(0) = 0$ and $z_i^0(0) = \gamma_i^0 = 0$ for all $i \in \mathcal{S}$. Then, we apply 10 SQP iterations to warm-start dSQP before continuing with the first NMPC step.

[4] `https://github.com/PREDICT-EPFL/holohover`

References

Aboudonia, A., A. Eichler, F. Cordiano, G. Banjac, and J. Lygeros (2022a). "Distributed Model Predictive Control With Reconfigurable Terminal Ingredients for Reference Tracking". In: *IEEE Transactions on Automatic Control* 67.11, pp. 6263–6270.

Aboudonia, A., A. Eichler, and J. Lygeros (2020). "Distributed Model Predictive Control with Asymmetric Adaptive Terminal Sets for the Regulation of Large-scale Systems". In: *IFAC-PapersOnLine* 53.2. 21st IFAC World Congress, pp. 6899–6904.

Aboudonia, A., A. Martinelli, N. Hoischen, and J. Lygeros (2022b). "Reconfigurable Plug-and-play Distributed Model Predictive Control for Reference Tracking". In: *2022 IEEE 61st Conference on Decision and Control (CDC)*, pp. 1130–1135.

Abughalieh, K. M. and S. G. Alawneh (2019). "A Survey of Parallel Implementations for Model Predictive Control". In: *IEEE Access* 7, pp. 34348–34360.

Allgöwer, F., J. Borges de Sousa, J. Kapinski, P. Mosterman, J. Oehlerking, P. Panciatici, M. Prandini, A. Rajhans, P. Tabuada, and P. Wenzelburger (2019). "Position paper on the challenges posed by modern applications to cyber-physical systems theory". In: *Nonlinear Analysis: Hybrid Systems* 34, pp. 147–165.

Allibhoy, A. and J. Cortés (2021). "Data-Based Receding Horizon Control of Linear Network Systems". In: *IEEE Control Systems Letters* 5.4, pp. 1207–1212.

Alonso, C. A., J. S. Li, J. Anderson, and N. Matni (2023). "Distributed and Localized Model-Predictive Control–Part I: Synthesis and Implementation". In: *IEEE Transactions on Control of Network Systems* 10.2, pp. 1058–1068.

Andersson, J. A. E., J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl (2019). "CasADi: A Software Framework for Nonlinear Optimization and Optimal Control". In: *Mathematical Programming Computation* 11.1, pp. 1–36.

Behrunani, V. N., H. Cai, P. Heer, R. S. Smith, and J. Lygeros (2024). "Distributed Multi-Horizon Model Predictive Control for Network of Energy Hubs". In: *Control Engineering Practice* 147, p. 105922.

Bernal Neira, D. E., C. D. Laird, L. R. Lueg, S. M. Harwood, D. Trenev, and D. Venturelli (2024). "Utilizing Modern Computer Architectures to Solve Mathematical Optimization Problems: A Survey". In: *Computers & Chemical Engineering* 184, p. 108627.

Bertsekas, D. P. (2016). *Nonlinear Programming*. 3rd ed. Athena Scientific.

Bertsekas, D. P. and J. N. Tsitsiklis (1989). *Parallel and Distributed Computation: Numerical Methods*. Vol. 23. Englewood Cliffs, NJ: Prentice Hall.

Bestler, A. and K. Graichen (2019). "Distributed Model Predictive Control for Continuous-Time Nonlinear Systems Based on Suboptimal ADMM". In: *Optimal Control Applications and Methods* 40.1, pp. 1–23.

Bhanderi, A., A. Rajashekhar, P. Vemana, F. Greiwe, C. Hams, T. Harnisch, M. Kaupmann, and A. Alhanafi (2021). *Optimization and Control via Time Sensitive and Software Defined Networking*. Tech. rep. TU Dortmund University.

Bitlislioğlu, A., I. Pejcic, and C. Jones (2017). "Interior Point Decomposition for Multi-Agent Optimization". In: *IFAC-PapersOnLine* 50.1, pp. 233–238.

Boggs, P. T. and J. W. Tolle (1995). "Sequential Quadratic Programming". In: *Acta numerica* 4, pp. 1–51.

Boyd, S., N. Parikh, E. Chu, B. Peleato, and J. Eckstein (2011). "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers". In: *Foundations and Trends in Machine Learning* 3.1, pp. 1–122.

Braun, P. (2016). "Hierarchical distributed optimization and predictive control of a smart grid". PhD thesis. University of Bayreuth.

Braun, P. and L. Grüne (2018). "Verteilte Optimierung: Anwendungen in der Modellprädiktiven Regelung". In: *at-Automatisierungstechnik* 66.11, pp. 939–949.

Braun, P., L. Grüne, C. M. Kellett, S. R. Weller, and K. Worthmann (2020). "Towards Price-Based Predictive Control of a Small-Scale Electricity Network". In: *International Journal of Control* 93.1, pp. 40–61. eprint: `https://doi.org/10.1080/00207179.2017.1339329`.

Bullo, F. (2024). *Lectures on Network Systems*. 1.7. Kindle Direct Publishing.

Bullo, F., J. Cortés, and S. Martínez (2009). *Distributed Control of Robotic Networks: A Mathematical Approach to Motion Coordination Algorithms*. Princeton Series in Applied Mathematics. Princeton University Press.

Burk, D., A. Völz, and K. Graichen (2021a). "A Modular Framework for Distributed Model Predictive Control of Nonlinear Continuous-Time Systems (GRAMPC-D)". In: *Optimization and Engineering* 23, pp. 771–795.

Burk, D., A. Völz, and K. Graichen (2021b). "Experimental Validation of the Open-Source DMPC Framework GRAMPC-D applied to the Remotely Accessible Robotarium". In: *Proceedings of the IEEE International Conference on Mechatronics and Automation (ICMA)*, pp. 442–447.

Burk, D., A. Völz, and K. Graichen (2021c). "Towards Asynchronous ADMM for Distributed Model Predictive Control of Nonlinear Systems". In: *2021 European Control Conference (ECC)*, pp. 1957–1962.

Camacho, E. F. and C. Bordons (2007). *Model Predictive Control*. Springer London.

Carron, A., D. Saccani, L. Fagiano, and M. N. Zeilinger (2023). "Multi-agent Distributed Model Predictive Control with Connectivity Constraint". In: *IFAC-PapersOnLine* 56.2. 22nd IFAC World Congress, pp. 3806–3811.

Chanfreut, P., A. Sánchez-Amores, J. M. Maestre, and E. F. Camacho (2021a). "Distributed Model Predictive Control based on Dual Decomposition with Neural-Network-based Warm Start". In: *2021 European Control Conference (ECC)*, pp. 1969–1974.

Chanfreut, P., J. M. Maestre, and E. F. Camacho (2021b). "A Survey on Clustering Methods for Distributed and Networked Control Systems". In: *Annual Reviews in Control* 52, pp. 75–90.

Chanfreut, P., J. M. Maestre, D. Krishnamoorthy, and E. F. Camacho (2023). "ALADIN-Based Distributed Model Predictive Control with Dynamic Partitioning: An Application to Solar Parabolic Trough Plants". In: *2023 62nd IEEE Conference on Decision and Control (CDC)*, pp. 8376–8381.

Christofides, P. D., R. Scattolini, D. Muñoz de la Peña, and J. Liu (2013). "Distributed Model Predictive Control: A Tutorial Review and Future Research Directions". In: *Computers & Chemical Engineering* 51, pp. 21–41.

Cohen, G. (1977). "On an Algorithm of Decentralized Optimal Control". In: *Journal of mathematical analysis and applications* 59.2, pp. 242–259.

Cohen, G. (1978). "Optimization by Decomposition and Coordination: A Unified Approach". In: *IEEE Transactions on Automatic Control* 23.2, pp. 222–232.

Cole, D., S. Shin, F. Pacaud, V. M. Zavala, and M. Anitescu (2023). "Exploiting GPU/SIMD Architectures for Solving Linear-Quadratic MPC Problems". In: *2023 American Control Conference (ACC)*, pp. 3995–4000.

Conte, C., C. N. Jones, M. Morari, and M. N. Zeilinger (2016). "Distributed Synthesis and Stability of Cooperative Distributed Model Predictive Control for Linear Systems". In: *Automatica* 69, pp. 117–125.

Conte, C., T. Summers, M. N. Zeilinger, M. Morari, and C. N. Jones (2012). "Computational Aspects of Distributed Optimization in Model Predictive Control". In: *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pp. 6819–6824.

Curtis, F. E., T. C. Johnson, D. P. Robinson, and A. Wächter (2014). "An Inexact Sequential Quadratic Optimization Algorithm for Nonlinear Optimization". In: *SIAM Journal on Optimization* 24.3, pp. 1041–1074.

Darivianakis, G., A. Eichler, and J. Lygeros (2020). "Distributed Model Predictive Control for Linear Systems With Adaptive Terminal Sets". In: *IEEE Transactions on Automatic Control* 65.3, pp. 1044–1056.

de Jong, T. and M. Lazar (2023). "Energy-Based Cooperative Distributed MPC for Frequency Control in Microgrids". In: *IEEE Control Systems Letters* 7, pp. 3441–3446.

Dembo, R. S., S. C. Eisenstat, and T. Steihaug (1982). "Inexact Newton Methods". In: *SIAM J. Numer. Anal.* 19.2, pp. 400–408.

Deng, W. and W. Yin (2016). "On the Global and Linear Convergence of the Generalized Alternating Direction Method of Multipliers". In: *Journal of Scientific Computing* 66.3, pp. 889–916.

Diehl, M., H. G. Bock, J. P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer (2002a). "Real-Time Optimization and Nonlinear Model Predictive Control of Processes Governed by Differential-Algebraic Equations". In: *Journal of Process Control* 12.4, pp. 577–585.

Diehl, M., R. Findeisen, F. Allgower, H. G. Bock, and J. Schlöder (2003). "Stability of Nonlinear Model Predictive Control in the Presence of Errors due to Numerical Online Optimization". In:

42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475). Vol. 2, 1419–1424 Vol.2.

Diehl, M., R. Findeisen, and F. Allgöwer (2007). "A Stabilizing Real-Time Implementation of Nonlinear Model Predictive Control". In: *Real-Time PDE-Constrained Optimization*, pp. 25–52. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9780898718935.ch2.

Diehl, M., R. Findeisen, S. Schwarzkopf, I. Uslu, F. Allgöwer, H. G. Bock, E. D. Gilles, and J. P. Schlöder (2002b). "An Efficient Algorithm for Nonlinear Model Predictive Control of Large-Scale Systems Part I: Description of the Method (Ein effizienter Algorithmus für die nichtlineare prädiktive Regelung großer Systeme Teil I: Methodenbeschreibung)". In: *at - Automatisierungstechnik* 50.12, p. 557.

Doan, M. D., M. Diehl, T. Keviczky, and B. De Schutter (2017). "A jacobi decomposition algorithm for distributed convex optimization in distributed model predictive control". In: *IFAC-PapersOnLine* 50.1, pp. 4905–4911.

Dörfler, F. and F. Bullo (2012). "Synchronization and Transient Stability in Power Networks and Nonuniform Kuramoto Oscillators". In: *SIAM Journal on Control and Optimization* 50.3, pp. 1616–1642.

Du, X., Y. Xi, and S. Li (2001). "Distributed Model Predictive Control for Large-Scale Systems". In: *Proceedings of the 2001 American Control Conference. (Cat. No.01CH37148)*. Vol. 4, 3142–3143 vol.4.

Ebel, H. (2021). *Distributed Control and Organization of Communicating Mobile Robots: Design, Simulation, and Experimentation*. Shaker.

Ebel, H. and P. Eberhard (2021). "A Comparative Look at two Formation Control Approaches Based on Optimization and Algebraic Graph Theory". In: *Robotics and Autonomous Systems* 136, p. 103686.

Ebel, H. and P. Eberhard (2022). "Cooperative Transportation: Realizing the Promises of Robotic Networks Using a Tailored Software/Hardware Architecture". In: *at – Automatisierungstechnik* 70.4, pp. 378–388.

Ebel, H., M. Rosenfelder, and P. Eberhard (2024). "Cooperative Object Transportation with Differential-Drive Mobile Robots: Control and Experimentation". In: *Robotics and Autonomous Systems* 173, p. 104612.

El Fawal, H., D. Georges, and G. Bornard (1998). "Optimal Control of Complex Irrigation Systems via Decomposition-Coordination and the Use of Augmented Lagrangian". In: *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.98CH36218)*. Vol. 4, 3874–3879 vol.4.

Engelmann, A. (2020). "Distributed Optimization with Application to Power Systems and Control". PhD thesis. Karlsruhe: Karlsruhe Institute of Technology (KIT).

Engelmann, A. and T. Faulwasser (2021). "Decentralized Conjugate Gradients With Finite-Step Convergence". In: *arXiv:2102.12311*.

Engelmann, A., Y. Jiang, H. Benner, R. Ou, B. Houska, and T. Faulwasser (2021a). "ALADIN-α—an Open-Source MATLAB Toolbox for Distributed Non-Convex Optimization". In: *Optimal Control Applications and Methods*, pp. 1–19. arXiv: 2006.01866.

Engelmann, A., Y. Jiang, B. Houska, and T. Faulwasser (2020). "Decomposition of Nonconvex Optimization via Bi-Level Distributed ALADIN". In: *IEEE Transactions on Control Network Systems* 7.4, pp. 1848–1858.

Engelmann, A., Y. Jiang, T. Mühlpfordt, B. Houska, and T. Faulwasser (2019). "Toward Distributed OPF Using ALADIN". In: *IEEE Transactions on Power Systems* 34.1, pp. 584–594.

Engelmann, A., M. Kaupmann, and T. Faulwasser (2022). "Essentially Decentralized Interior Point Methods for Optimization in Energy Networks". In: *Extended Abstracts of the 25th International Symposium on Mathematical Theory of Networks and Systems*, pp. 445 –448.

Engelmann, A., T. Mühlpfordt, Y. Jiang, B. Houska, and T. Faulwasser (2017). "Distributed AC Optimal Power Flow Using ALADIN". In: *IFAC-PapersOnLine*. Vol. 50, pp. 5536–5541.

Engelmann, A., G. Stomberg, and T. Faulwasser (2021b). "An Essentially Decentralized Interior Point Method for Control". In: *2021 60th IEEE Conference on Decision and Control (CDC)*, pp. 2414–2420.

Erseghe, T. (2014). "Distributed Optimal Power Flow Using ADMM". In: *IEEE Transactions on Power Systems* 29.5, pp. 2370–2380.

Erunsal, I. K., R. Ventura, and A. Martinoli (2024). "A Distributed Architecture for Onboard Tightly-Coupled Estimation and Predictive Control of Micro Aerial Vehicle Formations". In: *Distributed Autonomous Robotic Systems*. Cham: Springer Nature Switzerland, pp. 156–172.

Eschmann, H., H. Ebel, and P. Eberhard (2021). "Trajectory Tracking of an Omnidirectional Mobile Robot Using Gaussian Process Regression". In: *at – Automatisierungstechnik* 69.8, pp. 656–666.

Ferramosca, A., D. Limon, I. Alvarado, and E. F. Camacho (2013). "Cooperative Distributed MPC for Tracking". In: *Automatica* 49.4, pp. 906–914.

Ferreau, H. J., C. Kirches, A. Potschka, H. G. Bock, and M. Diehl (2014). "qpOASES: A Parametric Active-Set Algorithm for Quadratic Programming". In: *Mathematical Programming Computation* 6.4, pp. 327–363.

Fiacco, A. V. (1976). "Convergence Properties of Local Solutions of Sequences of Mathematical Programmign Problems in General Spaces". In: *Journal of Optimization Theory and Applications* 13, pp. 1–12.

Fiacco, A. V. (1983). *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming*. Ed. by R. Bellman. Academic Press.

Findeisen, R. (2006). *Nonlinear Model Predictive Control: A Sampled-Data Feedback Perspective*. Düsseldorf: Fortschr.-Ber. VDI Reihe 8 Nr. 1087, VDI Verlag.

Frank, M. and P. Wolfe (1956). "An Algorithm for Quadratic Programming". In: *Naval Research Logistics Quarterly* 3.1-2, pp. 95–110. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nav.3800030109.

Frank, S. and S. Rebennack (2016). “An Introduction to Optimal Power Flow: Theory, Formulation, and Examples”. In: *IIE Transactions* 48.12, pp. 1172–1197.

Gabay, D. and B. Mercier (1976). “A Dual Algorithm for the Solution of Nonlinear Variational Problems via Finite Element Approximation”. In: *Computers & Mathematics with Applications* 2.1, pp. 17–40.

Geiger, C. and C. Kanzow (2002). *Theorie und Numerik restringierter Optimierungsaufgaben*. Springer.

Giselsson, P., M. D. Doan, T. Keviczky, B. D. Schutter, and A. Rantzer (2013). “Accelerated Gradient Methods and Dual Decomposition in Distributed Model Predictive Control”. In: *Automatica* 49.3, pp. 829–833.

Giselsson, P. and A. Rantzer (2014). “On Feasibility, Stability and Performance in Distributed Model Predictive Control”. In: *IEEE Transactions on Automatic Control* 59.4, pp. 1031–1036.

Glowinski, R. and A. Marroco (1975). “Sur l’approximation, Par Éléments Finis d’ordre Un, et La Résolution, Par Pénalisation-Dualité d’une Classe de Problèmes de Dirichlet Non Linéaires”. In: *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique* 9, pp. 41–76.

Graichen, K. and A. Kugi (2010). “Stability and Incremental Improvement of Suboptimal MPC Without Terminal Constraints”. In: *IEEE Transactions on Automatic Control* 55.11, pp. 2576–2580.

Graichen, K. and B. Käpernick (2012). “A Real-Time Gradient Method for Nonlinear Model Predictive Control”. In: *Frontiers of Model Predictive Control*. Ed. by T. Zheng. Rijeka: InteChOpen. Chap. 1.

Grancharova, A., I. Valkova, N. Hvala, and J. Kocijan (2023). “Distributed Predictive Control Based on Gaussian Process Models”. In: *Automatica* 149, p. 110807.

Gros, S. (2014). “A Distributed Algorithm for NMPC-based Wind Farm Control”. In: *53rd IEEE Conference on Decision and Control*, pp. 4844–4849.

Gros, S., M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl (2020). “From Linear to Nonlinear MPC: Bridging the Gap via the Real-Time Iteration”. In: *International Journal of Control* 93.1, pp. 62–80.

Groß, D. and O. Stursberg (2013). “On the convergence rate of a jacobi algorithm for cooperative distributed MPC”. In: *Proc. 52nd IEEE Conference on Decision and Control*, pp. 1508–1513.

Grüne, L. and J. Pannek (2017). *Nonlinear Model Predictive Control: Theory and Algorithms*. Second. Springer International Publishing.

Guanetti, J., Y. Kim, and F. Borrelli (2018). “Control of Connected and Automated Vehicles: State of the Art and Future Challenges”. In: *Annual Reviews in Control* 45, pp. 18–40.

Han, D.-R. (2022). “A Survey on Some Recent Developments of the Alternating Direction Method of Multipliers”. In: *Journal of the Operations Research Society of China* 10, pp. 1–52.

Havlena, V., P. Trnka, and B. Sheridan (2014). “Management of Complex Water Networks”. In: *The Impact of Control Technology, 2nd Edition*. Ed. by T. Samad and A. Annaswamy. IEEE Control Systems Society.

Hentzelt, S., A. Klingler, and K. Graichen (2014). “Experimental Results for Distributed Model Predictive Control Applied to a Water Distribution System”. In: *2014 IEEE International Symposium on Intelligent Control (ISIC)*, pp. 1100–1106.

Hong, T., P. Pinson, Y. Wang, R. Weron, D. Yang, and H. Zareipour (2020). “Energy Forecasting: A Review and Outlook”. In: *IEEE Open Access Journal of Power and Energy* 7, pp. 376–388.

Horn, R. A. and C. R. Johnson (2013). *Matrix Analysis*. 2nd edition. Cambridge University Press.

Hours, J.-H. and C. N. Jones (2016). “A Parametric Nonconvex Decomposition Algorithm for Real-Time and Distributed NMPC”. In: *IEEE Transactions on Automatic Control* 61.2, pp. 287–302.

Houska, B., J. Frasch, and M. Diehl (2016). “An Augmented Lagrangian Based Algorithm for Distributed Nonconvex Optimization”. In: *SIAM J. Optim.* 26.2, pp. 1101–1127.

Houska, B. and Y. Jiang (2021). “Distributed Optimization and Control with ALADIN”. In: *Recent Advances in Model Predictive Control: Theory, Algorithms, and Applications*. Ed. by T. Faulwasser, M. A. Müller, and K. Worthmann. Cham: Springer International Publishing, pp. 135–163.

Houska, B. and J. Shi (2022). “Distributed MPC with ALADIN—A Tutorial”. In: *2022 American Control Conference (ACC)*, pp. 358–363.

Huang, A. S., E. Olson, and D. C. Moore (2010). “LCM: Lightweight Communications and Marshalling”. In: *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Taipei, pp. 4057–4062.

Huber, H., D. Burk, and K. Graichen (2022). “Comparison of Sensitivity-Based and ADMM-Based DMPC Applied to Building Automation”. In: *2022 IEEE Conference on Control Technology and Applications (CCTA)*, pp. 546–553.

Hult, R., M. Zanon, S. Gros, and P. Falcone (2016). “Primal Decomposition of the Optimal Coordination of Vehicles at Traffic Intersections”. In: *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, pp. 2567–2573.

Hult, R., M. Zanon, G. Frison, S. Gros, and P. Falcone (2020). “Experimental Validation of a Semi-Distributed Sequential Quadratic Programming Method for Optimal Coordination of Automated Vehicles at Intersections”. In: *Optimal Control Applications and Methods* 41.4, pp. 1068–1096. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/oca.2592`.

Hult, R., M. Zanon, S. Gros, and P. Falcone (2022). “A Semidistributed Interior Point Algorithm for Optimal Coordination of Automated Vehicles at Intersections”. In: *IEEE Transactions on Control Systems Technology* 30.5, pp. 1977–1989.

IBM (2009). IBM ILOG CPLEX Optimizer Version 12.1.

IBM (2024). IBM ILOG CPLEX Optimizer Version 22.1.1.0, https://www.ibm.com/de-de/products/ilog-cplex-optimization-studio/cplex-optimizer. Accessed on 24.10.2024.

Intel (2024). Intel oneMKL Pardiso Version 2024.0, https://www.intel.com/content/www/us/en/docs/onemkl/developer-reference-c/2024-0/onemkl-pardiso-parallel-direct-sparse-solver-iface.html. Accessed on 08.11.2024.

Jia, D. and B. Krogh (2001). "Distributed Model Predictive Control". In: *Proceedings of the 2001 American Control Conference. (Cat. No.01CH37148)*. Vol. 4, 2767–2772 vol.4.

Jiang, Y., P. Sauerteig, B. Houska, and K. Worthmann (2021). "Distributed Optimization Using ALADIN for MPC in Smart Grids". In: *IEEE Transactions on Control Systems Technology* 29.5, pp. 2142–2152.

Kang, J., Y. Cao, D. P. Word, and C. D. Laird (2014). "An Interior-Point Method for Efficient Solution of Block-Structured NLP Problems Using an Implicit Schur-Complement Decomposition". In: *Computers & Chemical Engineering* 71, pp. 563–573.

Kanjanawanishkul, K. and A. Zell (2008a). "A Model-Predictive Approach to Formation Control of Omnidirectional Mobile Robots". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2771–2776.

Kanjanawanishkul, K. and A. Zell (2008b). "Distributed Model Predictive Control for Coordinated Path Following Control of Omnidirectional Mobile Robots". In: *2008 IEEE International Conference on Systems, Man and Cybernetics*, pp. 3120–3125.

Karvelis, P. (2024). *daviolinplot*. github.com/povilaskarvelis/DataViz/releases/tag/v3.2.4. Accessed on 06 August 2024.

Katriniok, A., B. Rosarius, and P. Mähönen (2022). "Fully Distributed Model Predictive Control of Connected Automated Vehicles in Intersections: Theory and Vehicle Experiments". In: *IEEE Transactions on Intelligent Transportation Systems* 23.10, pp. 18288–18300.

Kayacan, E., E. Kayacan, H. Ramon, and W. Saeys (2014). "Distributed Nonlinear Model Predictive Control of an Autonomous Tractor–Trailer System". In: *Mechatronics* 24.8, pp. 926–933.

Kersbergen, B., T. van den Boom, and B. De Schutter (2016). "Distributed model predictive control for railway traffic management". In: *Transportation Research Part C: Emerging Technologies* 68, pp. 462–489.

Kim, K.-D. and P. R. Kumar (2012). "Cyber–Physical Systems: A Perspective at the Centennial". In: *Proceedings of the IEEE* 100.Special Centennial Issue, pp. 1287–1308.

Kögel, M. and R. Findeisen (2012). "Cooperative Distributed MPC Using the Alternating Direction Multiplier Method". In: *IFAC Proceedings Volumes* 45.15, pp. 445–450.

Köhler, J., M. A. Müller, and F. Allgöwer (2019). "Distributed Model Predictive Control—Recursive Feasibility Under Inexact Dual Optimization". In: *Automatica* 102, pp. 1–9.

Kozma, A., C. Conte, and M. Diehl (2015). "Benchmarking Large-Scale Distributed Convex Quadratic Programming Algorithms". In: *Optimization Methods and Software* 30.1, pp. 191–214.

Kurtz, F., G. Stomberg, M. B. Bandeira, J. Püttschneider, F. Greiwe, M. Kaupmann, C. Hams, T. Harnisch, M. O. Tay, A. Choudhary, et al. (2022). "Software-Defined Networking driven Time-Sensitive Networking for Mixed-Criticality Control Applications". In: *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 99–104.

Kuwata, Y., A. Richards, T. Schouwenaars, and J. P. How (2007). "Distributed Robust Receding Horizon Control for Multivehicle Guidance". In: *IEEE Transactions on Control Systems Technology* 15.4, pp. 627–641.

Lamnabhi-Lagarrigue, F., A. Annaswamy, S. Engell, A. Isaksson, P. Khargonekar, R. M. Murray, H. Nijmeijer, T. Samad, D. Tilbury, and P. Van den Hof (2017). "Systems & Control for the Future of Humanity, Research Agenda: Current and Future roles, Impact and Grand Challenges". In: *Annual Reviews in Control* 43, pp. 1–64.

Le, V.-A. and T. X. Nghiem (2020). "Gaussian Process Based Distributed Model Predictive Control for Multi-Agent Systems Using Sequential Convex Programming and ADMM". In: *Proc. of the IEEE Conference on Control Technology and Applications*. IEEE, pp. 31–36.

Lefebure, N., M. Khosravi, M. Hudoba de Badyn, F. Bünning, J. Lygeros, C. Jones, and R. S. Smith (2022). "Distributed Model Predictive Control of Buildings and Energy Hubs". In: *Energy and Buildings* 259, p. 111806.

Legat, B., O. Dowson, J. D. Garcia, and M. Lubin (2021). "MathOptInterface: A Data Structure for Mathematical Optimization Problems". In: *INFORMS Journal on Computing*.

Li, D. and B. De Schutter (2022). "Distributed Model-Free Adaptive Predictive Control for Urban Traffic Networks". In: *IEEE Transactions on Control Systems Technology* 30.1, pp. 180–192.

Li, G. and T. K. Pong (2015). "Global Convergence of Splitting Methods for Nonconvex Composite Optimization". In: *SIAM Journal on Optimization* 25.4, pp. 2434–2460.

Li, W. C. and L. T. Biegler (1988). "Process Control Strategies for Constrained Nonlinear Systems". In: *Industrial & Engineering Chemistry Research* 27.8, pp. 1421–1433. eprint: `https://doi.org/10.1021/ie00080a014`.

Limon, D., T. Alamo, F. Salas, and E. Camacho (2006). "On the stability of constrained MPC without terminal constraint". In: *IEEE Transactions on Automatic Control* 51.5, pp. 832–836.

Liu, D. and Q. Tran-Dinh (2020). "An Inexact Interior-Point Lagrangian Decomposition Algorithm with Inexact Oracles". In: *Journal of Optimization Theory and Applications* 185.3, pp. 903–926.

Liu, K., T. Liu, Z. Tang, and D. J. Hill (2019). "Distributed MPC-Based Frequency Control in Networked Microgrids With Voltage Constraints". In: *IEEE Transactions on Smart Grid* 10.6, pp. 6343–6354.

Liu, M. and J. Jian (2020). "An ADMM-Based SQP Method for Separably Smooth Nonconvex Optimization". In: *Journal of Inequalities and Applications* 2020.1, pp. 1–17.

Lu, L. (2015). "Separable Nonlinear Model Predictive Control via Sequential Quadratic Programming for Large-Scale Systems". In: *IFAC-PapersOnLine* 48.23, pp. 495–500.

Lubin, M., O. Dowson, J. Dias Garcia, J. Huchette, B. Legat, and J. P. Vielma (2023). “JuMP 1.0: Recent Improvements to a Modeling Language for Mathematical Optimization”. In: *Mathematical Programming Computation*.

Lucia, S., M. Kögel, and R. Findeisen (2015). “Contract-based Predictive Control of Distributed Systems with Plug and Play Capabilities”. In: *IFAC-PapersOnLine* 48.23. 5th IFAC Conference on Nonlinear Model Predictive Control NMPC 2015, pp. 205–211.

Luis, C. E. and A. P. Schoellig (2019). “Trajectory Generation for Multiagent Point-To-Point Transitions via Distributed Model Predictive Control”. In: *IEEE Robotics and Automation Letters* 4.2, pp. 375–382.

Luis, C. E., M. Vukosavljev, and A. P. Schoellig (2020). “Online Trajectory Generation With Distributed Model Predictive Control for Multi-Robot Motion Planning”. In: *IEEE Robotics and Automation Letters* 5.2, pp. 604–611.

Lunze, J. (2022). *Networked Control of Multi-Agent Systems*. 2nd Edition. Edition MoRa.

Ma, Y., G. Anderson, and F. Borrelli (2011). “A Distributed Predictive Control Approach to Building Temperature Regulation”. In: *Proceedings of the 2011 American Control Conference*, pp. 2089–2094.

Macenski, S., T. Foote, B. Gerkey, C. Lalancette, and W. Woodall (2022). “Robot Operating System 2: Design, Architecture, and Uses in the Wild”. In: *Science Robotics* 7.66, eabm6074.

Maciejowski, J. M. (2002). *Predictive Control with Constraints*. Harlow: Pearson Education.

Mayne, D. Q., J. B. Rawlings, C. V. Rao, and P. O. Scokaert (2000). “Constrained Model Predictive Control: Stability and Optimality”. In: *Automatica* 36.6, pp. 789–814.

Mehrez, M. W., T. Sprodowski, K. Worthmann, G. Mann, R. G. Gosine, J. K. Sagawa, and J. Pannek (2017). “Occupancy Grid Based Distributed MPC for Mobile Robots”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4842–4847.

Mercangöz, M. and F. J. Doyle (2007). “Distributed Model Predictive Control of an Experimental Four-Tank System”. In: *Journal of Process Control* 17.3. Special Issue ADCHEM 2006 Symposium, pp. 297–308.

Mesarovic, M. D., D. Macko, and Y. Takahara (1970). *Theory of Hierarchical, Multilevel, Systems*. Academic Press, New York.

Mills, A., A. Wills, and B. Ninness (2009). “Nonlinear Model Predictive Control of an Inverted Pendulum”. In: *2009 American Control Conference*, pp. 2335–2340.

Molzahn, D. K., F. Dörfler, H. Sandberg, S. H. Low, S. Chakrabarti, R. Baldick, and J. Lavaei (2017). “A Survey of Distributed Optimization and Control Algorithms for Electric Power Systems”. In: *IEEE Trans Smart Grid* 8.6, pp. 2941–2962.

Morini, B. (1999). “Convergence Behaviour of Inexact Newton Methods”. In: *Math. Comput.* 68.228, pp. 1604–1613.

Müller, M. A. and F. Allgöwer (2017). “Economic and Distributed Model Predictive Control: Recent Developments in Optimization-based Control”. In: *SICE Journal of Control, Measurement, and System Integration* 10.2, pp. 39–52.

Necoara, I. and J. A. K. Suykens (2009). “Interior-Point Lagrangian Decomposition Method for Separable Convex Optimization”. In: *Journal of Optimization Theory and Applications* 143.3, p. 567.

Necoara, I., D. N. Clipici, and S. Olaru (2013). “Distributed Model Predictive Control of Leader-Follower Systems Using an Interior Point Method with Efficient Computations”. In: *2013 American Control Conference*, pp. 1697–1702.

Necoara, I., C. Savorgnan, D. Q. Tran, J. Suykens, and M. Diehl (2009). “Distributed Nonlinear Optimal Control Using Sequential Convex Programming and Smoothing Techniques”. In: *Proc. of the 48th IEEE Conference on Decision and Control*, pp. 543–548.

Necoara, I. and J. A. K. Suykens (2008). “Application of a Smoothing Technique to Decomposition in Convex Optimization”. In: *IEEE Trans Aut Contr* 53.11, pp. 2674–2679.

Nedić, A., A. Olshevsky, and S. Wei (2018). “Decentralized Consensus Optimization and Resource Allocation”. In: *Large-Scale and Distributed Optimization*. Springer, pp. 247–287.

Nedić, A. and J. Liu (2018). “Distributed Optimization for Control”. In: *Annual Review of Control, Robotics, and Autonomous Systems* 1.Volume 1, 2018, pp. 77–103.

Nesterov, Y. (2018). *Lectures on Convex Optimization*. Vol. 137. Springer.

Nocedal, J. and S. Wright (2006). *Numerical Optimization*. Springer Science & Business Media, New York.

Oh, K.-K., M.-C. Park, and H.-S. Ahn (2015). “A Survey of Multi-Agent Formation Control”. In: *Automatica* 53, pp. 424–440.

Ohtsuka, T. (2004). “A Continuation/GMRES Method for Fast Computation of Nonlinear Receding Horizon Control”. In: *Automatica* 40.4, pp. 563–574.

Pacaud, F., S. Shin, M. Schanen, D. A. Maldonado, and M. Anitescu (2024). “Accelerating Condensed Interior-Point Methods on SIMD/GPU Architectures”. In: *Journal of Optimization Theory and Applications* 202, pp. 184–203.

Pakazad, S. K., A. Hansson, and M. S. Andersen (2014). “Distributed Interior-Point Method for Loosely Coupled Problems”. In: *IFAC Proceedings Volumes* 47.3, pp. 9587–9592.

Pierer von Esch, M., E. Nistler, A. Völz, and K. Graichen (2025a). “Sensitivity-Based Distributed NMPC: Experimental Results for a Levitating Planar Motion System”. In: *IEEE Transactions on Control Systems Technology*, pp. 1–0.

Pierer von Esch, M., A. Völz, and K. Graichen (2025b). “Sensitivity-Based Distributed Model Predictive Control for Non-Linear Systems Under Inexact Optimization”. In: *Optimal Control Applications and Methods* 46.4, pp. 1538–1558. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/oca.3277`.

Qin, S. and T. A. Badgwell (2003). “A survey of industrial model predictive control technology”. In: *Control Engineering Practice* 11.7, pp. 733–764.

Rajkumar, R. R., I. Lee, L. Sha, and J. Stankovic (2010). “Cyber-physical systems: the next computing revolution”. In: *Proceedings of the 47th Design Automation Conference*. DAC ’10. Anaheim, California: Association for Computing Machinery, 731–736.

Ramirez-Riberos, J. L., M. Pavone, E. Frazzoli, and D. W. Miller (2010). "Distributed Control of Spacecraft Formations via Cyclic Pursuit: Theory and Experiments". In: *Journal of Guidance, Control, and Dynamics* 33.5, pp. 1655–1669.

Rawlings, J. B., D. Q. Mayne, and M. Diehl (2019). *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing.

Razzanelli, M., E. Crisostomi, L. Pallottino, and G. Pannocchia (2020). "Distributed Model Predictive Control for Energy Management in a Network of Microgrids Using the Dual Decomposition Method". In: *Optimal Control Applications and Methods* 41.1, pp. 25–41. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/oca.2504.

Richards, A. and J. How (2005). "Implementation of Robust Decentralized Model Predictive Control". In: *AIAA Guidance, Navigation, and Control Conference and Exhibit*.

Richter, S. (2012). "Computational Complexity Certification of Gradient Methods for Real-Time Model Predictive Control". PhD thesis. ETH Zürich.

Richter, S., M. Morari, and C. N. Jones (2011). "Towards Computational Complexity Certification for Constrained MPC Based on Lagrange Relaxation and the Fast Gradient Method". In: *Proc. 50th IEEE Conference on Decision and Control and European Control Conference*. IEEE, pp. 5223–5229.

Robinson, S. M. (1974). "Perturbed Kuhn-Tucker points and rates of convergence for a class of nonlinear-programming algorithms". In: *Mathematical Programming* 7, pp. 1–16.

Rosenfelder, M., H. Ebel, and P. Eberhard (2022). "Cooperative Distributed Nonlinear Model Predictive Control of a Formation of Differentially-Driven Mobile Robots". In: *Robotics and Autonomous Systems* 150, p. 103993.

Rostami, R., G. Costantini, and D. Görges (2017). "ADMM-based distributed model predictive control: Primal and dual approaches". In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 6598–6603.

Savorgnan, C., C. Romani, A. Kozma, and M. Diehl (2011). "Multiple Shooting for Distributed Systems with Applications in Hydro Electricity Production". In: *Journal of Process Control* 21.5, pp. 738–745.

Scattolini, R. (2009). "Architectures for Distributed and Hierarchical Model Predictive Control – A Review". In: *Journal of Process Control* 19.5, pp. 723–731.

Schenk, O., K. Gärtner, W. Fichtner, and A. Stricker (2001). "PARDISO: A High-Performance Serial and Parallel Sparse Linear Solver in Semiconductor Device Simulation". In: *Future Generation Computer Systems* 18.1, pp. 69–78.

Schubiger, M., G. Banjac, and J. Lygeros (2020). "GPU Acceleration of ADMM for Large-Scale Quadratic Programming". In: *Journal of Parallel and Distributed Computing* 144, pp. 55–67.

Schwan, R., N. Schmid, E. Chassaing, K. Samaha, and C. Jones (2024). "On identifying the non-linear dynamics of a hovercraft using an end-to-end deep learning approach". In: *IFAC-PapersOnLine* 58.15. 20th IFAC Symposium on System Identification SYSID 2024, pp. 289–294.

Schwan, R., Y. Jiang, D. Kuhn, and C. N. Jones (2023). “PIQP: A Proximal Interior-Point Quadratic Programming Solver”. In: *2023 62nd IEEE Conference on Decision and Control (CDC)*, pp. 1088–1093.

Scokaert, P. O., D. Q. Mayne, and J. B. Rawlings (1999). “Suboptimal model predictive control (feasibility implies stability)”. In: *IEEE Transactions on Automatic Control* 44.3, pp. 648–654.

Shin, S., M. Anitescu, and F. Pacaud (2024). “Accelerating Optimal Power Flow with GPUs: SIMD Abstraction of Nonlinear Programs and Condensed-Space Interior-Point Methods”. In: *Electric Power Systems Research* 236, p. 110651.

Shin, S., M. Anitescu, and V. M. Zavala (2022). “Exponential Decay of Sensitivity in Graph-Structured Nonlinear Programs”. In: *SIAM Journal on Optimization* 32.2, pp. 1156–1183.

Shin, S., C. Coffrin, K. Sundar, and V. M. Zavala (2021). “Graph-Based Modeling and Decomposition of Energy Infrastructures”. In: *IFAC-PapersOnLine* 54.3. 16th IFAC Symposium on Advanced Control of Chemical Processes ADCHEM 2021, pp. 693–698.

Shorinwa, O. and M. Schwager (2023). “Distributed Target Tracking in Multi-Agent Networks via Sequential Quadratic Alternating Direction Method of Multipliers”. In: *2023 American Control Conference (ACC)*, pp. 341–348.

Siljak, D. D. (1991). *Decentralized Control of Complex Systems*. Academic Press.

Simon, D. (2006). *Optimal State Estimation*. John Wiley & Sons, Ltd.

Stellato, B., G. Banjac, P. Goulart, A. Bemporad, and S. Boyd (2020). “OSQP: An Operator Splitting Solver for Quadratic Programs”. In: *Mathematical Programming Computation*.

Stewart, B. T., A. N. Venkat, J. B. Rawlings, S. J. Wright, and G. Pannocchia (2010). “Cooperative Distributed Model Predictive Control”. In: *Systems & Control Letters* 59.8, pp. 460–469.

Stewart, B. T., S. J. Wright, and J. B. Rawlings (2011). “Cooperative Distributed Model Predictive Control for Nonlinear Systems”. In: *Journal of Process Control* 21.5, pp. 698–704.

Stomberg, G., H. Ebel, T. Faulwasser, and P. Eberhard (2023). “Cooperative Distributed MPC via Decentralized Real-Time Optimization: Implementation Results for Robot Formations”. In: *Control Engineering Practice* 138, p. 105579.

Stomberg, G., A. Engelmann, and T. Faulwasser (2021). “A Distributed Active Set Method for Model Predictive Control”. In: *IFAC-PapersOnLine* 54.3, pp. 263–268.

Stomberg, G., A. Engelmann, M. Diehl, and T. Faulwasser (2025a). “Decentralized Real-Time Iterations for Distributed NMPC”. In: *IEEE Transactions on Automatic Control*, pp. 1–16.

Stomberg, G., A. Engelmann, and T. Faulwasser (2022a). “A Compendium of Optimization Algorithms for Distributed Linear-Quadratic MPC”. In: *at - Automatisierungstechnik* 70.4, pp. 317–330.

Stomberg, G., A. Engelmann, and T. Faulwasser (2022b). “Decentralized Non-Convex Optimization via Bi-Level SQP and ADMM”. In: *Proceedings of the 61st IEEE Conference on Decision and Control (CDC)*, 273–278,
arXiv:2204.08786.

Stomberg, G., M. Raetsch, A. Engelmann, and T. Faulwasser (2025b). “Large problems are not necessarily hard: A case study on distributed NMPC paying off”. In: *2025 European Control Conference (ECC)*, pp. 2630–2637.

Stomberg, G., R. Schwan, A. Grillo, C. N. Jones, and T. Faulwasser (2025c). “Cooperative Distributed Model Predictive Control for Embedded Systems: Experiments with Hovercraft Formations”. In: *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11377–11383.

Summers, T. H. and J. Lygeros (2012). “Distributed Model Predictive Consensus via the Alternating Direction Method of Multipliers”. In: *Proceedings of the 50th Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 79–84.

Sun, K. and X. A. Sun (2023). “A Two-Level Distributed Algorithm for Nonconvex Constrained Optimization”. In: *Computational Optimization and Applications* 84, pp. 609–649.

Sun, K. and X. A. Sun (2021). “A Two-Level ADMM Algorithm for AC OPF With Global Convergence Guarantees”. In: *IEEE Transactions on Power Systems* 36.6, pp. 5271–5281.

Tang, W. and P. Daoutidis (2019). “Distributed nonlinear model predictive control through accelerated parallel ADMM”. In: *2019 American Control Conference (ACC)*, pp. 1406–1411.

Tang, W. and P. Daoutidis (2023). “Fast and Stable Nonconvex Constrained Distributed Optimization: the ELLADA Algorithm”. In: *Optimization and Engineering* 23, pp. 259–301.

Themelis, A. and P. Patrinos (2020). “Douglas–Rachford Splitting and ADMM for Nonconvex Optimization: Tight Convergence Results”. In: *SIAM Journal on Optimization* 30.1, pp. 149–181.

Trnka, P., J. Pekař, and V. Havlena (2011). “Application of Distributed MPC to Barcelona Water Distribution Network”. In: *IFAC Proceedings Volumes* 44.1. 18th IFAC World Congress, pp. 9139–9144.

Ulbrich, M. and S. Ulbrich (2012). *Nichtlineare Optimierung*. Birkhäuser.

Van Parys, R. and G. Pipeleers (2017). “Distributed MPC for Multi-Vehicle Systems Moving in Formation”. In: *Robotics and Autonomous Systems* 97, pp. 144–152.

Vargas, S., H. M. Becerra, and J.-B. Hayet (2022). “MPC-based Distributed Formation Control of Multiple Quadcopters with Obstacle Avoidance and Connectivity Maintenance”. In: *Control Engineering Practice* 121, p. 105054.

Velasquez, M. A., J. Barreiro-Gomez, N. Quijano, A. I. Cadena, and M. Shahidehpour (2019). “Distributed Model Predictive Control for Economic Dispatch of Power Systems with High Penetration of Renewable Energy Resources”. In: *International Journal of Electrical Power & Energy Systems* 113, pp. 607–617.

Venkat, A. N., I. A. Hiskens, J. B. Rawlings, and S. J. Wright (2008). “Distributed MPC strategies with application to power system automatic generation control”. In: *IEEE Transactions on Control Systems Technology* 16.6, pp. 1192–1206.

Venkat, A. N., J. B. Rawlings, and S. J. Wright (2005). “Stability and Optimality of Distributed Model predictive Control”. In: *Proc. 44th IEEE Conference on Decision and Control*, pp. 6680–6685.

Wächter, A. and L. T. Biegler (2006). “On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming”. In: *Mathematical Programming* 106.1, pp. 25–57.

Wang, M., I. McInerney, B. Stellato, S. Boyd, and H. K.-H. So (2023). “RSQP: Problem-specific Architectural Customization for Accelerated Convex Quadratic Optimization”. In: *Proceedings of the 50th Annual International Symposium on Computer Architecture*. ISCA ’23. Association for Computing Machinery.

Wang, Y. and C. Manzie (2022). “Robust Distributed Model Predictive Control of Linear Systems: Analysis and Synthesis”. In: *Automatica* 137, p. 110141.

Wang, Y., W. Yin, and J. Zeng (2019). “Global Convergence of ADMM in Nonconvex Nonsmooth Optimization”. In: *Journal of Scientific Computing* 78.1, pp. 29–63.

Wolf, I. J. and W. Marquardt (2016). “Fast NMPC Schemes for Regulatory and Economic NMPC–A Review”. In: *Journal of Process Control* 44, pp. 162–183.

Word, D. P., J. Kang, J. Akesson, and C. D. Laird (2014). “Efficient Parallel Solution of Large-Scale Nonlinear Dynamic Optimization Problems”. In: *Computational Optimization and Applications* 59.3, pp. 667–688.

Yang, W. H. and D. Han (2016). “Linear Convergence of the Alternating Direction Method of Multipliers for a Class of Convex Optimization Problems”. In: *SIAM Journal on Numerical Analysis* 54.2, pp. 625–640.

Yashtini, M. (2021). “Multi-block Nonconvex Nonsmooth Proximal ADMM: Convergence and Rates Under Kurdyka–Łojasiewicz Property”. In: *Journal of Optimization Theory and Applications* 190, pp. 966–998.

Yu, L., A. Goldsmith, and S. Di Cairano (2017). “Efficient Convex Optimization on GPUs for Embedded Model Predictive Control”. In: *Proceedings of the General Purpose GPUs*. GPGPU-10. Austin, TX, USA: Association for Computing Machinery, 12–21.

Zanelli, A. (2021). “Inexact Methods for Nonlinear Model Predictive Control: Stability, Application, and Software”. PhD thesis. University of Freiburg.

Zanelli, A., Q. Tran-Dinh, and M. Diehl (2021). “A Lyapunov Function for the Combined System-Optimizer Dynamics in Inexact Model Predictive Control”. In: *Automatica* 134, p. 109901.

Zanon, M., S. Gros, H. Wymeersch, and P. Falcone (2017). “An Asynchronous Algorithm for Optimal Vehicle Coordination at Traffic Intersections”. In: *IFAC-PapersOnLine* 50.1. 20th IFAC World Congress, pp. 12008–12014.

Zavala, V. M. and L. T. Biegler (2009). “The Advanced-Step NMPC Controller: Optimality, Stability and Robustness”. In: *Automatica* 45.1, pp. 86–93.

Zavala, V. M., C. D. Laird, and L. T. Biegler (2008). “Interior-Point Decomposition Approaches for Parallel Solution of Large-Scale Nonlinear Parameter Estimation Problems”. In: *Chemical Engineering Science* 63.19, pp. 4834–4845.

Zeilinger, M. N., Y. Pu, S. Riverso, G. Ferrari-Trecate, and C. N. Jones (2013). “Plug and Play Distributed Model Predictive Control Based on Distributed Invariance and Optimization”. In: *52nd IEEE Conference on Decision and Control*, pp. 5770–5776.