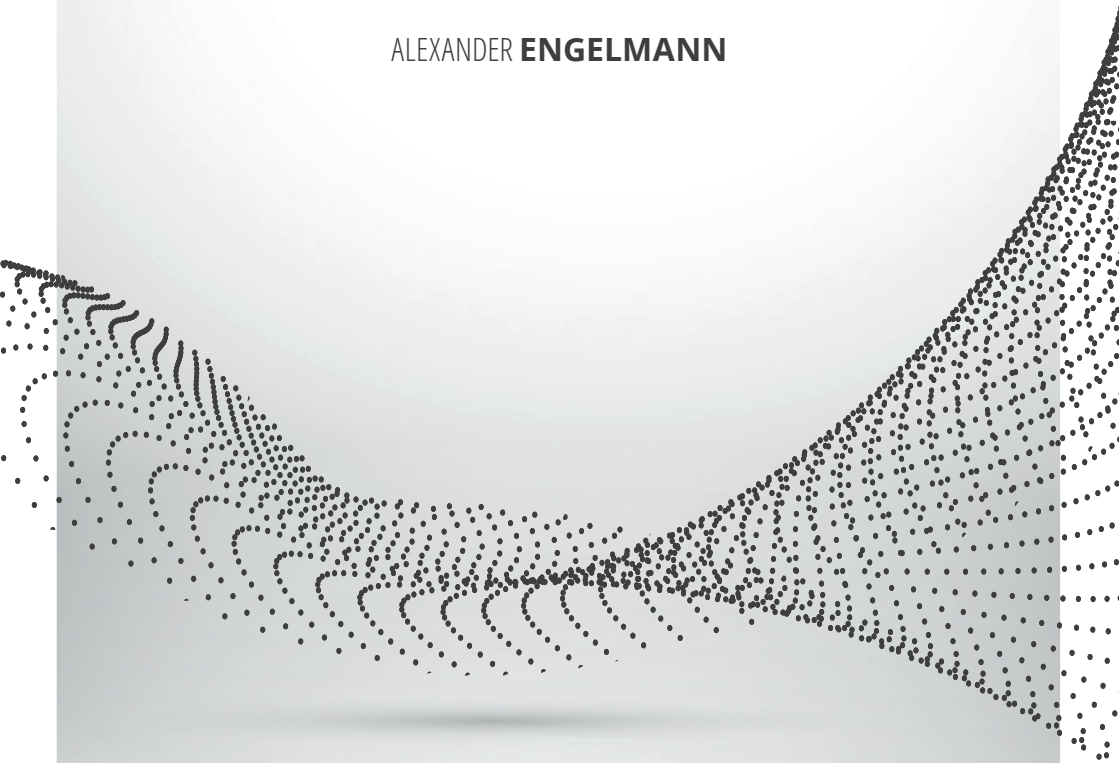
ALEXANDER **ENGELMANN**



DISTRIBUTED OPTIMIZATION WITH **APPLICATION** TO **POWER SYSTEMS** AND **CONTROL**

KIT Scientific Publishing

Alexander Engelmann

Distributed Optimization with Application to Power Systems and Control

Distributed Optimization with Application to Power Systems and Control

by
Alexander Engelmann

KIT Scientific Publishing

Karlsruher Institut für Technologie
Institut für Automation und angewandte Informatik

Distributed Optimization with Application
to Power Systems and Control

Zur Erlangung des akademischen Grades eines Doktor-Ingenieurs
von der KIT-Fakultät für Informatik des Karlsruher Instituts für
Technologie (KIT) genehmigte Dissertation

von Alexander Engelmann, M.Sc.

Tag der mündlichen Prüfung: 21. Oktober 2020
Referenten: Prof. Dr.-Ing. Timm Faulwasser
Korreferenten: Prof. Dr.-Ing. Veit Hagenmeyer
Prof. Colin Neil Jones, PhD

Impressum

Scientific
Publishing

Karlsruher Institut für Technologie (KIT)
KIT Scientific Publishing
Straße am Forum 2
D-76131 Karlsruhe

KIT Scientific Publishing is a registered trademark
of Karlsruhe Institute of Technology.
Reprint using the book cover is not allowed.

www.ksp.kit.edu

Print on Demand 2022 – Gedruckt auf FSC-zertifiziertem Papier

ISBN 978-3-7315-1180-9
DOI 10.5445/KSP/1000144792

Abstract

In many engineering domains, systems are composed of partially independent subsystems—power systems are composed of distribution and transmission systems, teams of robots are composed of individual robots, and chemical process systems are composed of vessels, heat exchangers and reactors. Often, these subsystems should reach a common goal such as satisfying a power demand with minimum cost, flying in a formation, or reaching an optimal set-point. At the same time, limited information exchange is desirable—for confidentiality reasons but also due to communication constraints. Moreover, a fast and reliable decision process is key as applications might be safety-critical.

Mathematical optimization techniques are among the most successful tools for controlling systems optimally with feasibility guarantees. Yet, they are often centralized—all data has to be collected in one central and computationally powerful entity. Methods from distributed optimization control the subsystems in a distributed or decentralized fashion, reducing or avoiding central coordination. These methods have a long and successful history. Classical distributed optimization algorithms, however, are typically designed for convex problems. Hence, they are only partially applicable in the above domains since many of them lead to optimization problems with non-convex constraints.

This thesis develops one of the first frameworks for distributed and decentralized optimization with non-convex constraints. Based on the Augmented Lagrangian Alternating Direction Inexact Newton (ALADIN) algorithm, a bi-level distributed ALADIN framework is presented, solving the coordination step of ALADIN in a decentralized fashion. This framework can handle various decentralized inner algorithms, two of which we develop here: a decentralized variant of the Alternating Direction Method of Multipliers (ADMM) and a novel decentralized Conjugate Gradient algorithm. Decentralized conjugate gradient is to the best of our knowledge the first decentralized algorithm with a guarantee of convergence to the exact solution in a finite number of iterates. Sufficient conditions for fast local convergence of bi-level ALADIN are

derived. Bi-level ALADIN strongly reduces the communication and coordination effort of ALADIN and preserves its fast convergence guarantees. We illustrate these properties on challenging problems from power systems and control, and compare performance to the widely used ADMM.

The developed methods are implemented in the open-source MATLAB toolbox ALADIN-α—one of the first toolboxes for decentralized non-convex optimization. ALADIN-α comes with a rich set of application examples from different domains showing its broad applicability. As an additional contribution, this thesis provides new insights why state-of-the-art distributed algorithms might encounter issues for constrained problems.

Deutsche Kurzfassung

In vielen Anwendungen bestehen Systeme aus einer Vielzahl von Subsystemen. Elektrische Energiesysteme bestehen aus Transportsystemen und Verteilsystemen, Roboter Teams bestehen aus einzelnen Robotern und chemische Prozesssysteme bestehen aus einzelnen Kesseln, Wärmetauschern und Reaktoren. Oftmals arbeiten diese Subsysteme auf ein gemeinsames Ziel hin, wie zum Beispiel die möglichst kostengünstige Deckung eines Energiebedarfs, das Fliegen in einer bestimmten Formation oder die kostenoptimale Ansteuerung eines Arbeitspunktes. Dabei ist ein möglichst geringer Informationsaustausch wünschenswert—sei es aus Vertraulichkeitsgründen oder auch aus Gründen limitierter Datenübertragungskapazität. Zusätzlich ist ein schneller und zuverlässiger Entscheidungsprozess essentiell—besonders im Falle sicherheitskritischer Anwendungen.

Methoden der mathematischen Optimierung gehören zu den erfolgreichsten Werkzeugen für die optimale Steuerung von Systemen mit Garantien. Noch sind diese Verfahren oftmals zentralisiert—alle Daten werden in einer zentralen, koordinierenden Einheit mit hoher Rechenleistung gesammelt. Methoden der verteilten Optimierung steuern Subsysteme auf verteilte oder dezentrale Weise unter Reduktion oder Vermeidung zentraler Koordination. Diese Methoden haben eine lange und erfolgreiche Historie. Klassische verteilte Optimierungsverfahren wurden in den meisten Fällen für konvexe Probleme entwickelt. Damit sind sie jedoch nur teilweise auf Probleme in den oben erwähnten Domänen anwendbar, da viele von ihnen auf Optimierungsprobleme mit nichtkonvexen Nebenbedingungen führen.

Die vorliegende Arbeit entwickelt eine der ersten Verfahrensklassen für die verteilte und dezentrale Optimierung unter nichtkonvexen Nebenbedingungen. Basierend auf dem Augmented Lagrangian Alternating Direction Inexact Newton (ALADIN) Algorithmus wird ein zweistufiges ALADIN Verfahren präsentiert, welches den Koordinationsschritt ALADIN's dezentral löst. Diese Verfahrensklasse ist in der Lage mit verschiedenen inneren dezentrale Ver-

fahren umzugehen, wobei zwei solcher Verfahren vorgestellt werden: eine dezentrale Variante des Alternating Direction of Multipliers Method (ADMM) Algorithmus und eine dezentrale Variante des konjugierte Gradienten Verfahrens. Das dezentrale konjugierte Gradienten Verfahren ist nach unserem Kenntnisstand das erste dezentrale Verfahren mit einer Konvergenzgarantie zur exakten Lösung eines Koordinationsproblems in einer endlichen Anzahl von Schritten. Hinreichende Bedingungen für die schnelle lokale Konvergenz des zweistufigen ALADIN Verfahrens werden hergeleitet. Zusätzlich zu seinen schnellen Konvergenzeigenschaften reduziert das zweistufige Verfahren den Kommunikations- und Koordinationsbedarf ALADIN's. Wir verdeutlichen diese Eigenschaften anhand herausfordernder Anwendungsbeispiele, die in elektrischen Energiesystemen sowie in der Optimalsteuerung auftreten, und vergleichen die Leistungsfähigkeit zu dem weitverbreiteten ADMM Verfahren.

Die entwickelten Verfahren sind in der quelloffenen MATLAB Toolbox ALADIN-α implementiert—eine der ersten Toolboxen für die dezentrale nichtkonvexe Optimierung. ALADIN-α verfügt über vielfältige Anwendungsbeispiele kommend aus verschiedenen Domänen, welches ihre breite Anwendbarkeit unterstreicht. Als zusätzlichen Beitrag dokumentiert diese Arbeit neue Einsichten über die Gründe, warum aktuelle verteilte Optimierungsverfahren in manchen Fällen Schwierigkeiten beim Lösen von Optimierungsprobleme unter Nebenbedingungen haben.

Acknowledgment

I would like to thank the many individuals, who supported me in the last three years of my PhD journey. Without your continuous support, this thesis would certainly not have been possible.

First of all, I would like to thank my supervisor Prof. Dr. Timm Faulwasser for his continuous excellent supervision and support, the many opportunities for international scientific exchange and also for his trust and the freedom in being able to pursue side projects, many of which worked out very well. Furthermore, I would like to thank Prof. Dr. Veit Hagenmeyer, who supported me especially in the last months of my PhD time. I would also like to thank Prof. Conlin Jones for serving as an excellent examiner of my thesis and also Prof. Snelting, Prof. Beckert and Prof. Abeck for participating in my committee.

Furthermore, I would like to thank all my colleagues from the Institute for Automation and applied Informatics. Riccardo, Till and Alex, you were and are not only colleagues to me, but also close friends. With you, Riccardo, it was always fun to discuss about many things far beyond research (but also related to that). We were the "power engineers" in our group, which led to many interesting and fruitful discussions with the "math and control engineers". With Till, my office mate, I remember countless exciting discussions about mathematical subjects far beyond the scope of my thesis. I really enjoyed this time and would also like to thank you for your support in difficult situations. Alex, also with you I enjoyed many discussions about research, politics and many other subjects. Furthermore, I would like to thank many other colleagues from IAI for the good relationship in the past years—both professionally and privately (in alphabetic order): Benedikt, Bernadette, Claudia, Clemens, Dominique, Frederik, Heiko, Henrieke, Jürgen, Lutz, Philipp, Sabine and Shadi. It was always fun to work with you and I really enjoyed the positive atmosphere in the insitute—keep it on! Supervising students was a highlight for me: thank you Xinlinang, Ruchuan and Sina for our time spent together!

Furthermore, I would like to thank my scientific partners—especially the group of Prof. Boris Houska from ShanghaiTech. I really enjoyed the time we spent together—especially at my visit in Shanghai. Thank you Boris for your continuous excellent support, feedback and encouragement. Yuning, we started our PhD essentially at the same time and had a close collaboration ever since. I enjoyed your creativity and thank you for the many projects we successfully finished together. I would also like to thank you, Xu, for exploring many interesting new research directions. Furthermore, I would like to thank my partners from the SmiLES project, thank you (in alphabetic order) Benedikt, Edmund, Isabelle, Oliver, Tue, Thomas, and Zhichao for all the good times spent together in various workshops and after-workshop events.

The next few sentences dedicated to the most important perople in my life are written in german. Liebe Mama, lieber Papa, ich möchte euch für eure Unterstützung danken, ich kann immer auf euch zählen, das ist unendlich wertvoll für mich. Juliane, Simon, Oma, Opa, ihr seid sehr wichtig in meinem Leben—ihr erdet mich und steht mir immer zur Seite. Auch viele meiner Freunde haben mich bei vielen Schwierigkeiten auf meiner Reise sehr unterstützt—hier möchte mich vor allen Dingen bei dir, Nico, bedanken. Du bist der Beste, du warst immer da, wenn es die ein oder andere Konfliktreiche Situation zu umschiffen galt und hast mir eine sehr wichtige äußere Perspektive auf die Dinge gegeben. Ich konnte immer auf dich zählen. Danke dafür!

Oktober 2020 Alexander Engelmann

Acronyms

ALADIN	Augmented Lagrangian Alternating Direction Inexact Newton
ADMM	Alternating Direction Method of Multipliers
SQP	Sequential Quadratic Programming
IP	Interior Point
NLP	Nonlinear Program
QP	Quadratic Program
KKT	Karush-Kuhn-Tucker
LICQ	Linear Independence Constraint Qualification
CG	Conjugate Gradient
DSG	Distributed Subgradient
OPF	Optimal Power Flow

Frequently Used Symbols

Distributed optimization

x, z	primal variables
λ, γ, μ	dual variables
$\mathcal{X}$	feasible set
$\mathcal{R}$	set of subsystems
$\mathcal{A}$	set of active constraints
$\mathcal{S}, (\mathcal{S}_\mu)$	set of (strongly) convex functions
$\iota_{\mathcal{X}}$	indicator function to the set $\mathcal{X}$
ρ	penalty parameter
Σ	weighting matrix
$(\cdot)^k$	outer iteration index
$(\cdot)^n$	inner iteration index
$(\cdot)^\star$	optimal value

Power systems

y	admittance of a branch
Y	bus admittance matrix
G, B	real/imaginary part of Y
θ, v	voltage angle/magnitude
s, p, q	apparent/active/reactive power
$\mathcal{N}$	set uf buses (nodes)
$\{\mathcal{N}_i\}$	partition of $\mathcal{N}$
$\mathcal{E}$	set of branches (edges)
$(\cdot)_k$	quantities related bus k
$(\cdot)_{kl}$	quantities related to the branch from bus k to bus l
$(\cdot)^{\mathrm{g}}, (\cdot)^{\mathrm{d}}, (\cdot)^{\mathrm{s}}$	generation/demand/storage

Content

1 Introduction

In many engineering domains, one can observe a trend towards systems composed of interconnected subsystems coordinating towards a common goal. Examples range from power systems, which collaborate to satisfy power demands, via robotics, collaborating to fulfill a certain task, to chemical engineering, where reactors collaborate to produce a certain product in an optimal fashion. Because of confidentiality reasons and communication constraints, this collaboration should be achieved subject to limited information exchange and limited central coordination while being fast and reliable.

One particular example for such systems are electricity grids. Electricity grids are typically composed of smaller grids, each of which is operated by one system operator. All together, they aim at a cheap and safe energy supply. To achieve this goal, it is usually necessary to exchange grid models and consumption data. Exchanging this data is, however, often problematic because of confidentiality reasons. Moreover, avoiding a single point of failure (e.g. a central coordinator) is also important for security reasons.

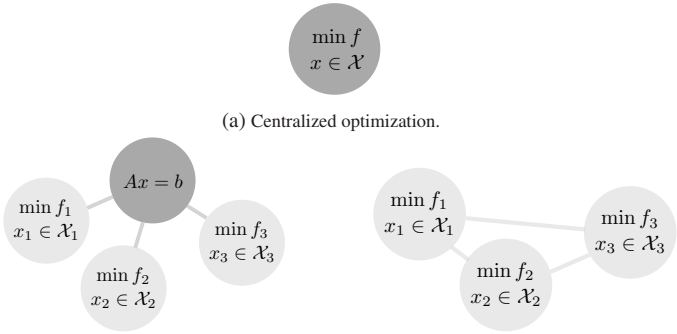
This thesis is about operating such systems by means of mathematical optimization techniques.

Employing mathematical optimization

Mathematical optimization is one of the most successful techniques for controlling systems automatically and optimally towards a goal. Mathematical optimization problems are typically formulated as

$$\min_x f(x) \quad \text{subject to} \quad x \in \mathcal{X}, \tag{1.1}$$

where $f : \mathcal{X} \to \mathbb{R}$ is called the *objective function*, $\mathcal{X}$ is called the *constraint set* or *feasible set* and x is called the *decision vector*. The objective function



(a) Centralized optimization.

(b) Distributed optimization.

(c) Decentralized optimization.

Figure 1.1: Distributed optimization, decentralized and centralized optimization.

f encodes the goal (such as minimizing the cost of power generation in an electricity grid) and the constraint set $\mathcal{X}$ captures the underlying physical model (such as an electricity grid model) and engineering limitations (such as maximum generator outputs). Problems in form of (1.1) are typically solved by *centralized* optimization algorithms in the sense that the data of all subsystems is collected in one central entity. This entity solves the problem and broadcasts the solution to the subsystems (Figure 1.1a).

Centralized optimization is often not desirable because of information aggregation in a single entity, which implies the existence of a single point of failure. Hence, techniques from *distributed* optimization are important. Distributed optimization means to shift computation mainly to the subsystems with a coordinating entity (Figure 1.1b).[1] Distributed optimization algorithms typically require problems, where f and $\mathcal{X}$ have a *special structure*. One common structure is

$$\min_x \sum_{i \in \mathcal{R}} f_i(x_i) \quad \text{subject to} \quad x_i \in \mathcal{X}_i, \;\; i \in \mathcal{R} \quad \text{and} \quad \sum_{i \in \mathcal{R}} A_i x_i = b. \tag{1.2}$$

[1] Note that, although the terminology of "distributed algorithms" might be used slightly different in computer science, there is an intimate connection between these two fields. We briefly comment on this interconnection in Appendix D.

Here, the optimization problem is distributed among a set of subsystems $\mathcal{R} = \{1, \ldots, R\}$, where each subsystem has its own objective function f_i (for example encoding the generator cost of one system operator), its own decision vector x_i (encoding voltages and powers in the region controlled by the system operator) and its own constraint set $\mathcal{X}_i$ (containing the grid model and engineering constraints of one region). Coupling between the subsystems is considered via a so-called consensus constraint $\sum_{i \in \mathcal{R}} A_i x_i = b$. This consensus constraint encodes the need for compatible physical values at the interconnection points between two neighboring subsystems (such as a matching power import/export between neighboring system operators).

While distributed optimization reduces the amount of central coordination, it still requires a coordinator. The appealing promise of *decentralized* optimization algorithms is to overcome the need for this coordinator, i.e. they avoid central coordination entirely and exchange information directly between neighbored subsystems (Figure 1.1c).

Although distributed and especially decentralized optimization algorithms are promising candidates for controlling systems with limited information exchange, classical distributed and decentralized algorithms are typically designed for *convex* problems. In many practical applications, however, models are non-linear leading to *non-convex* constraint sets. Hence, classical distributed and decentralized algorithms are often not applicable—at least not with convergence guarantees. Moreover, classical algorithms are often slow in practice.

In view of the above, the main research question of this thesis is as follows:

How to design efficient decentralized optimization algorithms for problems with non-convex constraints under limited information exchange and guaranteeing fast convergence?

Outline and contributions

Next, we outline the content of this thesis and highlight contributions of each chapter.

Chapter 2—Basics of Distributed Optimization

Chapter 2 briefly recalls basics of distributed optimization. We start with the fundamentals of nonlinear programming such as optimality conditions and prominent nonlinear programming algorithms. We continue by recalling two popular classical distributed optimization algorithms serving as building blocks for the algorithms we derive in Chapter 4 and serving as benchmark algorithms for numerical tests in Chapter 5. We recall the basic form of the Augmented Lagrangian Alternating Direction Inexact Newton (ALADIN) algorithm, which builds the foundation of the bi-level ALADIN algorithm, which we will develop in Chapter 4. Chapter 2 concludes with new insights why classical distributed algorithms might exhibit severe issues for constrained problems. We show that ALADIN is able to overcome these limitations by its more advanced coordination step, which explicitly considers constraint and curvature information.

Chapter 3—A survey on Distributed Optimization

In Chapter 3, we provide a literature review on distributed and decentralized optimization algorithms from different fields. The review is new in this form, since most literature reviews consider distributed algorithms from one particular community only. We conclude Chapter 3 by showing that most of the approaches so far have difficulties especially for problems with non-convex constraints.

Chapter 4—Bi-level Distributed ALADIN

Chapter 4 presents the main conceptual contribution of this thesis: a bi-level distributed ALADIN framework combining basic ALADIN with condensing techniques from nonlinear programming and an inner decentralization layer.[2] By using condensing techniques, we significantly lower coordination and com-

[2] For the sake of readability, we will simply write bi-level ALADIN in the following. Note that we do not make any connection to bi-level optimization problems in the sense of nested optimization problems [CMS07].

munication requirements of ALADIN. Decentralization is achieved by solving the coordination step by means of decentralized inner algorithms. To this end, we propose a decentralized variant of ADMM and, as an alternative, a novel decentralized variant of the conjugate gradient algorithm for solving the coordination step of ALADIN. Decentralized conjugate gradient has the advantage of converging in a finite number of iterations, while ADMM achieves at most linear convergence. We derive both algorithms based on a new sparsity encoding technique. As these two inner algorithms are both iterative in nature, they introduce inexactness to the coordination step of bi-level ALADIN. We show that bi-level ALADIN, nonetheless, preserves the fast convergence guarantees of basic ALADIN if the inaccuracy in the coordination step of ALADIN decays fast enough.

In summary, the contributions of Chapter 4 are

- one of the first decentralized optimization frameworks for problems with non-convex constraints (bi-level ALADIN);
- a reduced-space variant of ALADIN, lowering its communication and coordination requirements;
- a decentralized inner ADMM algorithm tailored to the coordination step of bi-level ALADIN;
- a new decentralized inner conjugate gradient algorithm solving the coordination step of ALADIN in a finite number of iterations;
- a new sparsity-encoding technique unifying the derivation of decentralized inner algorithms;
- a convergence analysis of bi-level ALADIN under inexact coordination.

The results of this chapter appeared in [Eng+20]. The sparsity encoding technique is not published so far.

Chapter 5—Application to Power Systems

Chapter 5 applies ALADIN and bi-level ALADIN to Optimal Power Flow (OPF) problems—an important problem class from power systems. We start

by reviewing relevant literature on distributed OPF and conclude that state-of-the art algorithms such as ADMM often lack convergence guarantees and often exhibits slow convergence. As an alternative, we propose ALADIN variants and compare ALADIN, condensed ALADIN and bi-level ALADIN to ADMM as a state-of-the-art algorithms for distributed OPF. We provide a numerical comparison in terms of convergence, coordination and communication for practically relevant cases. Moreover, we show that ADMM leads to feasible but not necessarily to optimal solutions in case of high penalization parameters in combination with a feasible initial guess. This is particularly important in the context of OPF as the combination of these two assumptions is sometimes used in the distributed OPF literature.

In summary, the contributions of Chapter 4 are

- one of the first algorithms for solving distributed OPF problems with guarantees;
- a performance comparison of basic ALADIN, bi-level ALADIN to ADMM as prominent state-of-the-art algorithm;
- a mathematical and numerical comparison in terms of communication, coordination and convergence of the above algorithms;
- a mathematical and numerical convergence analysis of ADMM for high penalization parameters in combination with a feasible initial guess;
- an analysis of reasons for limited accuracy in case of using bi-level ALADIN with ADMM as inner algorithm.

The results of this chapter have been published in [Eng+19b; Eng+17; Eng+20; EF18], where [Eng+19b] mainly focuses on numerical tests on cases up to 300 buses. [EF18] analyzes the convergence of ADMM in case of high penalization parameters in combination with a feasible initial point.

Chapter 6—The ALADIN-α toolbox

Chapter 6 presents an open-source toolbox, ALADIN-α, implementing ALADIN, bi-level ALADIN and the Alternating Direction Method of Multipliers (ADMM) in a unified framework. This toolbox is designed to be modular hav-

ing rapid-prototyping of distributed and decentralized optimization algorithms in mind. ALADIN-α supports advanced features such as parallel computing and parametric programming. The toolbox comes with a rich set of examples from different engineering fields highlighting its broad applicability. Specifically, we illustrate possible applications of ALADIN-α beyond power systems on a numerical example from distributed optimal control. ALADIN-α is one of the first toolboxes for decentralized non-convex optimization.

More explicitly, the contributions of ALADIN-α can be summarized as

- one of the first toolboxes for distributed and decentralized optimization with non-convex constraints;
- a modular framework for rapid-prototyping of distributed and decentralized optimization;
- a unified interface making comparisons to state-of-the-art algorithms such as ADMM easy;
- a wide range of numerical application examples from optimal control, power systems, sensor networks, machine learning and robotics.

The results of this chapter have been submitted for publication in [Eng+20].

Chapter 7 summarizes this thesis and proposes promising directions of future work.

2 Basics of Distributed Optimization

This chapter briefly reviews basics of *nonlinear programming*, where f and $\mathcal{X}$ in problem (1.1) are described by continuously differentiable functions. We also describe widely-used distributed optimization algorithms, where the derivation of dual decomposition is based on [Boy+11] and [BT89]. The derivation of ADMM comes from [Hou+17] with the difference that here we explicitly consider equality constraints. The optimality conditions are mainly from [NW06] and the illustrating examples in Section 3.4 are novel.

2.1 Basics of nonlinear programming

Nonlinear programs (NLPs) are special cases of (1.1) in form of

$$\min_{x \in \mathbb{R}} f(x) \tag{2.1a}$$

$$\text{subject to} \quad x \in \mathcal{X} = \{x \in \mathbb{R}^{n_x} \mid g(x) = 0,\ h(x) \leq 0\} \tag{2.1b}$$

where $g : \mathbb{R}^{n_x} \to \mathbb{R}^{n_g}$ and $h : \mathbb{R}^{n_x} \to \mathbb{R}^{n_h}$ are called *equality* and *inequality* constraints and $x \in \mathbb{R}^{n_x}$ is called the *decision vector* of (2.1).[1] Here, f, g and h are assumed to be sufficiently often continuously differentiable. Let us endow the constraint set $\mathcal{X}$ with a norm $\|\cdot\|$ yielding a normed space $(\mathcal{X}, \|\cdot\|)$. Then we can define minimizers to problem (2.1) as follows.

Definition 1 (Minimizers) We call $x^\star \in X$ a *local minimizer* to problem (1.1), if $f(x^\star) \leq f(x)$ for all $x \in \mathcal{B}_r(x^\star) \cap \mathcal{X}$ with $\mathcal{B}_r(\bar{x}) = \{x \mid \|x - \bar{x}\| < r\}$, $r > 0$. If $f(x^\star) \leq f(x)$ for *all* $x \in \mathcal{X}$, we call $x^\star$ a *global minimizer* to problem (1.1).

[1] Other special cases are *integer* or *mixed-integer* problems for example, where $\mathcal{X} \subseteq \mathbb{R}^{n_x} \times \mathbb{Z}^{n_z}$; or *infinite-dimensional* problems, where $\mathcal{X}$ is a subset of a function space. We refer to [Grö18] for an overview many important classes of practical relevance.

If the above inequalities hold strictly, we call $x^\star$ a *strict* local minimizer respectively *strict* global minimizer. We call $\mathcal{A}(\bar{x}) := \{i \in \{1, \dots, n_h\} \mid h_i(x) = 0\}$ the set of *active* or *binding* inequality constraints at a point $\bar{x}$. We call problem (2.1) *feasible*, if $\mathcal{X}$ is non-empty.

2.1.1 Optimality conditions

How can one find minimizers to (2.1)? One approach is to derive *optimality conditions* which (when solved) provide solution candidates to (2.1). Many numerical algorithms rely on the first-order necessary Karush-Kuhn-Tucker (KKT) conditions, which are a set of non-linear equations and non-linear inequalities providing candidate solution points to (2.1). To ensure that the KKT consitions hold true at a minimizer, one relies on constraint qualifications. Here we recall the strongest constraint qualification—the *linear independence constraint qualification* (LICQ).[2]

Definition 2 (Linear independence constraint qualification) Let $\bar{x}$ be a feasible point to (2.1). LICQ is said to hold at $\bar{x}$, if $\nabla h_i(\bar{x})$ and $\nabla g_i(\bar{x})$ are linearly independent for all $i \in \mathcal{A}(\bar{x})$ and for all $i \in \{1, \dots, n_g\}$.

Now, we are ready to state the KKT conditions.

[2] Weaker constraint qualifications exist, for example the Mangasarian-Fromovitz Constraint Qualification (MFCQ) or the Abadie Constraint Qualification. Moreover, there are first-order optimality conditions which do not require constraint qualifications, for example the Fritz-John optimality conditions [GGT04; BSS13]. However, these conditions are more difficult to evaluate and thus we stick with the KKT conditions and LICQ which is a common approach in the literature.

Theorem 1 (First-order optimality KKT conditions) Let f, g and h be continuously differentiable and let $x^\star$ be a minimizer to (2.1) for which LICQ holds. Then there exist unique vectors $\gamma^\star \in \mathbb{R}^{n_g}$ and $\mu^\star \in \mathbb{R}^{n_h}$ such that

$$0 = \nabla f(x^\star) + \sum_{i\in\{1,\dots,n_g\}} \gamma_i^\star \nabla g_i(x^\star) + \sum_{j\in\{1,\dots,n_h\}} \mu_i^\star \nabla h_i(x^\star), \tag{2.2a}$$

$$0 = g(x^\star), \tag{2.2b}$$

$$0 \geq h(x^\star), \tag{2.2c}$$

$$\mu^\star \geq 0, \tag{2.2d}$$

$$0 = \mu_i^\star h_i(x^\star) \quad \text{for all} \quad i \in \{i, \dots, n_h\}. \tag{2.2e}$$

We call $(x^\star, \gamma^\star, \mu^\star)$ a *KKT point*, if it satisfies the KKT conditions (2.2).

Next, we recall sufficient conditions for optimality. Consider the *Lagrangian* function to problem (2.1)

$$\mathcal{L}(x, \gamma, \mu) := f(x) + \gamma^\top g(x) + \mu^\top h(x),$$

where $\gamma^\top = (\gamma_1, \dots, \gamma_{n_g})$ and $\mu^\top = (\mu_1, \dots, \mu_{n_h})$.

Theorem 2 (Second-order sufficient condition [NW06, Thm 12.6]) Let $x^\star$ be a KKT point where LICQ holds. Morover, let $\nabla^2_{xx}\mathcal{L}(x^\star, \gamma^\star, \mu^\star)$ be positive definite on the subspace $\{x \in \mathbb{R}^{n_x} \mid \nabla g(x^\star)x = 0\}$. Then, $x^\star$ is a strict local minimizer to problem (2.1).[3]

Here, we denote the Jacobian matrix of g as $\nabla g(x) = \big(\nabla g_1(x), \dots, \nabla g_{n_g}(x)\big)^\top$. Many numerical algorithms rely on solving (2.2) for obtaining solution candidates to (2.1). The second-order sufficient conditions can then be used to distinguish minimizers from maximizers and saddle points. This forms the basis for two of the most popular algorithms for nonlinear programming: Sequential Quadratic Programming (SQP) and Interior Point (IP) methods.

[3] Note that the subspace assumption $\{x \in \mathbb{R}^{n_x} \mid \nabla g(x^\star)x = 0\}$ is slightly stronger than the one in the literature, where this subspace can be further reduced by means of the inequality constraint components h_i which are active at $x^\star$. However, this would make the presentation much more technical and would not give much additional generality for our purposes. For a more general version we refer to [NW06, Thm 12.6].

Observe that in the special case of equality constraints only, solving (2.2) reduces to solving a nonlinear system of equations which can be solved via standard Newton methods.

Convex problems

An important class of problems are *convex optimization problems*. Hence, let us recall basics of convex optimization.

Definition 3 (Convex set) A set $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ is called convex if the line between any two points $x_1, x_2 \in \mathcal{X}$ lies entirely in $\mathcal{X}$, i.e. $\theta\, x_1 + (1-\theta)\, x_2 \in \mathcal{X}$ for all $\theta \in (0 \;\; 1)$.

If the above inequality holds strictly for all $x_1, x_2 \in \mathcal{X}$, $x_1 \neq x_2$, $\mathcal{X}$ is called *strictly convex*.

Definition 4 (Convex function) A function $f : \mathcal{X} \mapsto \mathbb{R}$ is called convex if its epigraph is convex, i.e. if for all $x_1, x_2 \in \mathcal{X}$, we have $f(\theta\, x_1 + (1-\theta)\, x_2) \leq \theta\, f(x_1) + (1-\theta)\, f(x_2)$ for all $\theta \in (0 \;\; 1)$.

If the above inequality holds strictly for all $x_1 \neq x_2$, f is called *strictly convex*.

Definition 5 (Strongly convex function) If f is differentiable and $\big(\nabla f(x_1) - \nabla f(x_2)\big)^\top (x_1 - x_2) \geq \mu \, \|x_1 - x_2\|_2^2$ holds for all $x_1, x_2 \in \mathcal{X}$, f is called strongly convex with parameter μ.

We denote the set of convex functions with $\mathcal{S}$ and the set of strongly convex functions with parameter μ with $\mathcal{S}_\mu$. Note that any strongly convex function is strictly convex but not vice versa.

Convex *optimization problems* are special cases of (1.1), where f and $\mathcal{X}$ are convex. In case of nonlinear programming problems (2.1), this means that the inequality constraints h are convex and the equality constraints g are affine [Boy+04]. A nice property of convex problems is that their minima $f(x^\star)$ are unique (if they exist). Moreover, if f is strongly convex also the minimizer $x^\star$ is unique. In addition, the KKT conditions (2.2) are necessary *and* sufficient

for convex problems and thus one can omit the evaluation of the conditions of Theorem 2. Furthermore, specialized algorithms for convex optimization problems exist where some of them can be solved in polynomial time. Although we focus on non-convex optimization in this work, we will use these results from convex optimization in some of our derivations.

Next, we recall some basics from convex optimization which are essential to our later developments. In convex optimization, the *dual problem* (which is a maximization problem) to a (primal) minimization optimization problem is often useful since one can recover the solution to the primal problem from the solution to the dual problem under certain conditions. In many cases, the dual problem is considerable easier to solve than the primal problem—thus many algorithms focus on solving the dual problem. Recovering the solution to the primal problem from the dual is possible particularly in absence of a *duality gap*, for which we recall conditions next.

Consider the partial Lagrangian of (2.1) with respect to g, $\mathcal{L}(x, \gamma) = f(x) + \gamma^\top g(x)$. Then, the *dual function* to (2.1) is defined as

$$q(\gamma) := \inf_{x \in \mathcal{X}} \mathcal{L}(x, \gamma) = \inf_{x \in \mathcal{X}} f(x) + \gamma^\top g(x) \tag{2.3}$$

with $\mathcal{X} := \{x \in \mathbb{R}^{n_x} \mid h(x) \leq 0\}$. Moreover, the dual problem to (2.1) is

$$\sup_{\gamma} q(\gamma). \tag{2.4}$$

This leads to the following theorem.

Theorem 3 (Strong duality under Slater's condition [Ber99, Props 5.1.4, 5.1.5, 5.3.1]) Let f be convex on $\mathcal{X}$ and let g and h be convex, let (2.1) be feasible. Moreover, let some point $\bar{x}$ with $h(\bar{x}) < 0$ exist (Slater's condition). Then,

$$\sup_{\gamma} q(\gamma) = \inf_{g(x)=0, h(x) \leq 0} f(x),$$

i.e. there is no duality gap between (2.1) and (2.4). Moreover, the set of maximizers to (2.4), $\mathcal{D}$, is the same as the set of dual solutions to (2.1) and the set of primal solutions is given by the minimizers to

$$\min_{x \in \mathcal{X}} \mathcal{L}(x, \gamma^\star), \quad \gamma^\star \in \mathcal{D}. \tag{2.5}$$

Roughly speaking, this means that if the problem at hand is convex and Slater's condition holds, one can "replace" the original problem with it's dual. After solving the dual problem, a primal solution $x^\star$ can be recovered by means of (2.5).

Remark 1 (Slater's condition) Note that Slater assumption is standard in convex optimization and not overly restrictive. For special problem classes moreover, the assumptions can be relaxed. If h is affine for example, the strict feasibility condition can be relaxed to feasibility [Ber99, Prop 5.2.1]. Thus for feasible and convex quadratic programs for example, Slater's condition is always satisfied.

2.1.2 Sequential quadratic programming

There are four main branches for solving non-linear and possibly non-convex optimization problems in form of (2.1):

- gradient-based methods;
- sequential quadratic programming (SQP);
- interior point (IP) methods;
- augmented Lagrangian methods;

and combinations thereof. Gradient-based methods are typically used for large-scale problems, where second-order information is too expensive to evaluate. SQP and IP methods are very successful and exhibit fast convergence to local minimizers in many cases. However, each iteration is more costly compared with gradient-based methods. Augmented Lagrangian methods are often used for large-scale problems and as a building block for other algorithms (also within SQP). Here, we review the fundamental basics of SQP and augmented Lagrangian methods, because both are essential for understanding of the Augmented Lagrangian Alternating Direction Inexact Newton (ALADIN) algorithm. Gradient methods and interior point methods are not considered here—instead we refer to the textbooks [NW06; Ber99; Wri97; Nes13] for details.

SQP algorithms can be derived in two different ways. One is based on the idea of solving (2.1) by sequentially approximating the problem by a quadratic program (QP) where its name comes from. We recall the second way here, viewing SQP as solving the optimality conditions (2.2) by a Newton-type method. In absence of inequality constraints, the KKT conditions (2.2) for problem (2.1) read

$$F(x,\gamma) := \begin{pmatrix} \nabla f(x) + \gamma^\top \nabla g(x) \\ g(x) \end{pmatrix} \stackrel{!}{=} 0. \tag{2.6}$$

To solve this problem (and thus finding KKT points), let us apply a Newton-type method defined via the iteration

$$q^{k+1} = q^k - (M^k)^{-1} F(q^k), \tag{2.7}$$

where $q^k := (x^k, \gamma^k)$ and M^k is a non-singular approximation of $\nabla F(q^k)$. The Newton-type iteration (2.7) applied to (2.6) yields

$$\begin{pmatrix} B^k & \nabla g(x^k)^\top \\ \nabla g(x^k) & 0 \end{pmatrix} \begin{pmatrix} x^{k+1} - x^k \\ \gamma^{k+1} - \gamma^k \end{pmatrix} = \begin{pmatrix} \nabla f(x^k) + \gamma^{k\top} \nabla g(x^k) \\ g(x^k) \end{pmatrix}, \tag{2.8}$$

where B^k is an approximation of $\nabla^2_{xx} \mathcal{L}(x^k, \gamma^k)$.[4] In the next paragraph we will recall that if an SQP algorithm is initialized close enough to a local minimizer $(x^\star, \gamma^\star)$, the pair (x, γ) will converge to $(x^\star, \gamma^\star)$ at locally quadratic rate provided that $\nabla^2_{xx} \mathcal{L}(x, \gamma)$ is positive definite and LICQ holds at $(x^\star, \gamma^\star)$. SQP can be extended to problems with inequalities—we refer to [NW06, Chap. 18] for details.

Local convergence of Newton-type methods

Next, we give a proof of local convergence and convergence rates for the above Newton iteration (2.8). We consider *approximations* of the KKT matrix M^k instead of exact ones. These approximations are important for example if one

[4] Positive definiteness of B^k is essential for local convergence of Newton-type methods. These methods differ in the way they compute B^k, cf. [NW06, Ch 3.4].

would like to use Hessian approximations $B^k \approx \nabla^2_{xx}\mathcal{L}(x^k, \gamma^k)$ instead of exact Hessians $\nabla^2_{xx}\mathcal{L}(x^k, \gamma^k)$ when using Newton-type methods in context of SQP. When doing so, the local convergence rate depends also on the "quality" of these approximations. Local convergence of the iteration (2.8) is guaranteed by the following theorem.

Theorem 4 (Local convergence of Newton-type methods [Die16, Thm 8.1]) Suppose that $F(q^\star) = 0$ holds. Moreover, suppose that q^k is initialized close-enough to $q^\star$, i.e. with $\|q^k - q^\star\| < \frac{2(1-\gamma)}{\omega}$ where γ and ω are defined below. Then, the iterates generated by (2.8) converge to $q^\star$ and satisfy the convergence-rate estimate

$$\|q^{k+1} - q^\star\| \leq \left(\kappa + \frac{\omega}{2}\|q^k - q^\star\|\right)\|q^k - q^\star\|, \tag{2.9}$$

if there exists an $\omega < \infty$ and a $\kappa < 1$ such that

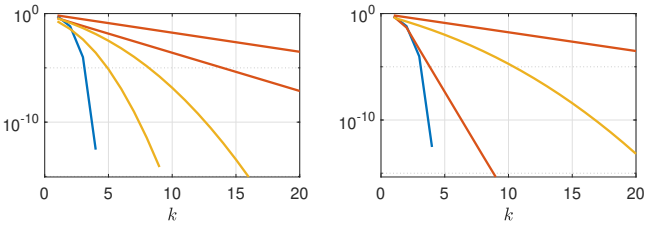
$$\left\|(M^k)^{-1}(M^k - \nabla F(q^k))\right\| \leq \kappa^k \leq \kappa, \tag{2.10a}$$

$$\left\|(M^k)^{-1}\left(\nabla F(q) - \nabla F(q^k)\right)\right\| \leq \frac{\omega}{2}\|q^k - q\|. \tag{2.10b}$$

Here the former inequality (2.10a) is called *compatibility condition* and the latter inequality (2.10b) is a *Lipschitz condition*. The proof of Theorem 4 is given in Appendix A.4 for the sake of completeness and because of its importance for the convergence analysis of ALADIN. Next, we provide examples for local convergence rates to provide some intuition on the effect of the two conditions (2.10a) and (2.10b) in Theorem 4.

Example 1 (Convergence rates of Newton-type methods) Assume that ∇F is Lipscitz continuous. Then, if we choose a non-singular M^k, the Lipschitz condition (2.10b) is always satisfied.

For checking the compatibility condition (2.10a), consider three variants of Jacobian approximations M^k: exact Jacobians with $M^k = \nabla F(q^k)$; damped Newton with $M^k = \nabla F(q^k) + \delta I$ for some $\delta > 0$; and damped Newton with vanishing damping $M^k = \nabla F(q^k) + 10^{-ck} I$ for some $c > 0$. For the first case, it immediately follows by (2.10a) that $\kappa = 0$. Thus, we have local quadratic convergence by (2.9) (cf. Section A.1 for a brief overview on convergence rate estimates).



(a) Typical convergence rates. (b) Linear conv. faster than superlinear.

Figure 2.1: Pure Newton (blue), damped-Newton (red) and damped-Newton with vanishing damping (yellow) for solving $\sin(x) = 0$.

For damped Newton, condition (2.10a) reads $\|(\nabla F(q^k) + \delta I)^{-1}(\nabla F(q^k) + \delta I - \nabla F(q^k))\| = \left\|(\nabla F(q^k) + \delta I)^{-1}\right\| \delta = \kappa$, where one can make κ arbitrarily small by choosing a small δ if $\nabla F(q^k)$ is non-singular. This yields linear convergence with arbitrary fast modulus κ.

For damped Newton with vanishing damping term we have $\kappa^k := \|(\nabla F(q^k) + 10^{-ck} I)^{-1}\| 10^{-ck}$ converging to zero for $k \to \infty$. Thus, if $\nabla F(q^k)$ is non-singular, one obtains superlinear convergence.

Example 2 (Linear convergence is not necessarily slow) Let us consider the problem of solving $F(x) = \sin(x) = 0$. Figure 2.1a shows the convergence of pure Newton's method (blue), damped Newton's method (red) with $\delta \in \{0.8,\ 2\}$ and damped-Newton with vanishing damping (yellow) and $c \in \{0.3,\ 0.1\}$. One can clearly see that by choosing appropriate δ and c, one can make convergence of the damped Newton variants almost as fast as the pure Newton's method. Thus, linear convergence does not necessarily mean that convergence is slow—it largely depends on the convergence modulus κ. This is especially relevant since damped-Newton methods for example typically converge at a quite fast linear rate if the damping parameter is sufficiently small. First-order methods such as ADMM, which we will introduce later, can be quite slow but both algorithms are guaranteed to converge linearly for certain problem classes.

Remark 2 (Singular $\nabla F(q^\star)$) Note that if $\nabla F(q^\star)$ is singular, the convergence rate is in general linear only—also with exact Jacobians $M^k = \nabla F(q^k)$. In that case, the Lipschitz condition (2.10b) is violated, since $\|(\nabla F(q^k))^{-1}\| \to \infty$ for

$q^k \to q^\star$. Thus, the second last step of the proof of Theorem 4 (Appendix A.4) can not be performed. In this case we only know that by continuity of ∇F, $\int_0^1 \left\|(M^k)^{-1}\left(\nabla F\left(q^\star + t(q^k - q^\star)\right) - \nabla F(q^k)\right)\right\| dt$ is bounded yielding linear convergence. As an example consider the problem $F(x) = x^2 = 0$ with $x^\star = 0$. Then the Newton iteration reads $x^{k+1} = x^k - (x^k)^2/(2x^k) = x^k/2$ and thus by taking norms $\|x^{k+1} - 0\| \leq 1/2 \|x^k - 0\|$. Notice that in this case, regularization procedures have to be considered as ∇F becomes increasingly ill conditioned close to $q^\star$ and thus factorization of M^k becomes problematic.

Remark 3 (Local vs. global convergence) Note that one distinguishes between local and global convergence in optimization algorithms. Local convergence characterizes the convergence behavior of algorithms when initialized close to a local minimizer. In order to reach this region of local convergence, typically globalization routines ar required. These routines take aim at driving the iterates into the region of local convergence and should then typically "deactivate". However, note that although one might guess that if an algorithm is globally convergent, it converges to a local minimizer from any initial point. This is unfortunately not the case. Typically it is only guaranteed to converge to a stationary point of a so-called merit function, which can for example be the augmented Lagrangian. Unfortunately, not all stationary points of the augmented Lagrangian are local minimizers [Ber99, Chap 4.3]. Especially in case of non-convex constraints, the situation of linearly dependent constraint linearizations can occur contradicting the assumption of linear independence of the constraints sometimes made for the global convergence analysis of nonlinear programming algorithms [BT95, Sec 4]. We give a concrete example what kind of difficulties can occur especially in context of non-convex constraints in Section 2.3.3.

2.1.3 Multiplier methods

Instead of working on the optimality conditions (2.2) directly and updating primal and dual variables in a simultaneous step, there is a different class of algorithms called *multiplier methods* [Ber82; Ber99]. There, the idea is to update primal and dual variables separately. This has advantages in several contexts—especially for developing distributed algorithms as in some cases the primal updates can be done in a distributed fashion under certain

conditions. There are two basic approaches in this class of algorithms: one working on the Lagrangian function and a second one working on the so-called *augmented Lagrangian* function which adds some kind of regularization term to the objective function without changing the minimizer. The augmented Lagrangian for (2.1) (in case of equality constraints only) is defined as

$$\mathcal{L}^\rho(x, \gamma) := f(x) + \gamma^\top g(x) + \frac{\rho}{2}\|g(x)\|_2^2.$$

Note that for $\rho = 0$, one recovers the standard Lagrangian. The method of multipliers now proceeds in two simple steps: First, we estimate a certain Lagrange multiplier λ^k. Then we minimize the Lagrangian/augmented Lagrangian with respect to x in a separate step. The resulting algorithm, summarized in Algorithm 1, is guaranteed to converge to a minimizer of (2.1) if the multiplier estimates are close enough to $\lambda^\star$. If we would estimate the multipliers correctly, multiplier methods converge in one step if ρ is large enough, cf. [Ber99, Sec 3.2.1]. Moreover, multiplier methods converge to a *global* minimizer of (2.1) for $\rho \to \infty$ [Ber99, Prop 4.2.1]. However, note that although this result is very appealing from an theoretical point of view, in practice driving ρ to ∞ is problematic due to the corresponding ill-conditioning of $\nabla^2 \mathcal{L}^\rho(x, \gamma)$ and the necessity to minimize $\mathcal{L}^\rho$ globally which is hard to achieve by practical solvers. If we choose the multiplier update to

Algorithm 1. General multiplier method

Initialization: γ^0, ρ
Repeat:

1) $x^{k+1} = \operatorname{argmin}_x \mathcal{L}^\rho(x, \gamma^k)$
2) $\gamma^{k+1} \leftarrow \gamma^k$

$$\gamma^{k+1} = \gamma^k + c^k g(x^k), \tag{2.11}$$

the method is called *method of multipliers*.

2.2 Distributed optimization algorithms

Now let us consider optimization problems with special structure leading to *distributed optimization algorithms*. Note that in the literature various forms of such structures are employed [ND11]. In many cases, they can be converted into each other. We give a brief overview on these structures in Appendix A.2. Here, we consider problems in so-called *affinely-coupled separable form*

$$\min_{x_i,\dots,x_R} \sum_{i\in\mathcal{R}} f_i(x_i) \tag{2.12a}$$

$$\text{subject to} \quad g_i(x_i) = 0, \qquad \forall i \in \mathcal{R}, \tag{2.12b}$$

$$h_i(x_i) \leq 0, \qquad \forall i \in \mathcal{R}, \tag{2.12c}$$

$$\sum_{i\in\mathcal{R}} A_i x_i = b, \tag{2.12d}$$

where $\mathcal{R} = \{1,\dots,N\}$ is the set of *agents* or *subsystems*, to each of which we assign an objective function $f_i : \mathbb{R}^{n_{xi}} \to \mathbb{R}$, equality constraints $g_i : \mathbb{R}^{n_{xi}} \to \mathbb{R}^{n_{gi}}$ and inequality constraints $h_i : \mathbb{R}^{n_{xi}} \to \mathbb{R}^{n_{hi}}$. We assume that all f_i, g_i and h_i are sufficiently often continuously differentiable. The agents are coupled trough (2.12d) which is called *consensus* or *coupling* constraint. Note that many practical problems in form of (2.1) can be reformulated in form of (2.12) by introducing auxilliary variables, cf. Appendix A.2. Next, we recall one of the earliest distributed optimization methods called *dual decomposition* [Eve63].

2.2.1 Dual decomposition

To simplify notation, we use the indicator function $\iota_{\mathcal{X}} : X \to \mathbb{R} \cup \{\infty\}$,

$$x \mapsto \begin{cases} 0 & \text{if} \quad x \in \mathcal{X} \\ \infty & \text{else} \end{cases}$$

in the subsequent analysis. With the indicator function we can reformulate (2.12) as

$$\min_{x_1,\dots,x_R} \sum_{i\in\mathcal{R}} \tilde{f}_i(x_i) \tag{2.13a}$$

$$\text{subject to} \quad \sum_{i\in\mathcal{R}} A_i x_i = b, \tag{2.13b}$$

with $\tilde{f}_i := f_i + \iota_{\mathcal{X}_i}$ and $\mathcal{X}_i := \{x_i \in \mathbb{R}^{n_{xi}} \mid g_i(x_i) = 0,\ h_i(x_i) \leq 0\}$. As the name indicates, dual decomposition is closely related to solving the *dual problem* to (2.13). Let us assume that the assumptions made in Theorem 3 (convexity, Slater's condition) hold true for problem (2.13). Then we can replace (2.13) by its dual problem

$$\sup_{\lambda} q(\lambda) = \sup_{\lambda} \min_{x\in\mathcal{X}} \sum_{i\in\mathcal{R}} \tilde{f}_i(x_i) + \lambda^\top \left(\sum_{i\in\mathcal{R}} A_i x_i - b \right). \tag{2.14}$$

One very simple way to solve (2.14) is to apply a gradient method. If f is strictly convex and $\mathcal{X}$ is compact, one can show that q is continuously differentiable and its gradient is given by

$$\nabla q(\lambda) = \sum_{i\in\mathcal{R}} A_i x_i(\lambda) - b,$$

where $x_i(\lambda)$ is computed by evaluating the dual function q [Ber99, Prop 6.1.1]. The gradient iteration for maximization of (2.14) then is

$$\lambda^{k+1} = \lambda^k + \alpha\, \nabla q(\lambda^k), \tag{2.15}$$

where $\alpha \in (0\ 1)$ is a stepsize parameter. Observe that q is separable, i.e we can write q as $q(\lambda) = \sum_{i\in\mathcal{R}} q_i(\lambda) - \lambda^\top b$, where $q_i(\lambda) = \min_{x_i\in\mathcal{X}_i} f_i(x_i) + \lambda^\top A_i x_i$, so for fixed λ, q can be evaluated *in parallel*. The resulting gradient method is called *dual decomposition* and summarized in Algorithm 2, where $\mathcal{L}_i(x_i, \lambda) := f_i(x_i) + \lambda^\top A_i x_i$. Note that dual decomposition is very effective when the minimization in the evaluation of the dual can be replaced by an explicit expression (e.g. when solving strictly convex QPs).

An advantage of dual decomposition is that it is very simple to implement. Moreover, if the A_is are sparse (i.e. there are many A_is with zero rows), it

Algorithm 2. Dual decomposition

Initialization: λ^0, $\{\alpha_k\}$

Repeat:

1) $x_i^{k+1} = \underset{x_i \in \mathcal{X}_i}{\operatorname{argmin}} \mathcal{L}_i(x_i, \lambda^k) \qquad \forall i \in \mathcal{R}$ (parallel)

2) $\lambda^{k+1} = \lambda^k + \alpha^k(\sum_{i \in \mathcal{R}} A_i x_i^{k+1} - b)$ (centralized)

is possible to evaluate step 2) of Algorithm 2 in a decentralized fashion as computing the components of λ^k involves summands of the subsystems with nonzero rows in A_i only. On the other hand, dual decomposition requires strict convexity of f and compactness of $\mathcal{X}$, or strong convexity of f as otherwise the minimizer in step 1) might not exist.[5] In turn this means that for general convex problems, dual decomposition might fail. Moreover, as dual decomposition is a gradient method with a fixed stepsize, convergence is in general sublinear only [Nes18, Thm 2.1.14].

2.2.2 Alternating Direction Method of Multipliers

To overcome its weaknesses (in particular the restriction to strictly/strongly convex problems), dual decomposition can be combined with the method of multipliers from Section 2.1.3 yielding the *Alternating Directions Method of Multipliers (ADMM)*. Herein, the idea is to use the *augmented Lagrangian* (instead of the Lagrangian) adding an additional convexification term rendering this algorithm applicable to convex but not necessarily strictly convex problems. Unfortunately the augmentation term in the first step of the method of multipliers jeopardizes the separability of $\mathcal{L}$ hindering evaluation of the dual function in parallel.[6] To overcome this issue, one usually constructs an augmented problem to (2.13a) introducing variable copies z of x and then executes

[5] This can be the case even for very simple problems. For example, consider a problem with affine cost. Then the problem in step 1) might be unbounded from below and dual decomposition would fail.

[6] As an alternative, one can also see ADMM as a dual ascent method, where one solves a modified but equivalent version of (2.13a) with a penalization term $\rho/2\| \sum_{i \in \mathcal{R}} A_i x_i - b\|$ in the objective [Boy+11].

the first step of the method of multipliers (Section 2.1.3) in an "alternating" or coordinate descent fashion recovering partial separability. We will now briefly recall ADMM for the augmented problem and then show how our problem (2.13a) fits into this setting.

The augmented problem in so called two-block form reads

$$\min_{x,z} \; f(x) + g(z) \tag{2.16}$$

$$\text{subject to} \;\; Cx + Dz = c. \tag{2.17}$$

For applying the method of multipliers, we consider the augmented Lagrangian

$$\mathcal{L}^{\rho}(x, z, \lambda) = f(x) + g(z) + \lambda^{\top}(Cx + Dz - c) + \frac{\rho}{2} \|Cx + Dz - c\|_2^2 . \tag{2.18}$$

We apply the method of multipliers (Algorithm 1) to problem (2.16) with one key difference: instead of performing a joint minimization with respect to x and z in step 1), we perform the minimization with respect to x and with respect to z in two separate steps. This yields ADMM shown in Algorithm 3.

Algorithm 3. Alternating Direction of Multipliers Method (ADMM)

Initialization: z^0, λ^0, ρ

Repeat:

1) $x^{k+1} = \operatorname{argmin}_x \mathcal{L}^{\rho}(x \;\; , z^k, \lambda^k)$
2) $z^{k+1} = \operatorname{argmin}_z \mathcal{L}^{\rho}(x^{k+1}, z \;\; , \lambda^k)$
3) $\lambda^{k+1} = \lambda^k + \rho(Cx^{k+1} + Dz^{k+1} - c)$

Now consider problem (2.13a). Let us write (2.13a) in two-block form by introducing variable copies $z_i = x_i$ for all $i \in \mathcal{R}$ yielding

$$\min_{x_1,\ldots,x_R,z_1,\ldots,z_R} \sum_{i \in \mathcal{R}} \tilde{f}_i(x_i) + \iota_C(z) \tag{2.19a}$$

$$\text{subject to} \quad A_i(x_i - z_i) = 0 \qquad \forall i \in \mathcal{R}, \tag{2.19b}$$

where $C := \{z \in \mathbb{R}^{n_x} \mid \sum_{i\in\mathcal{R}} A_i z_i = b\}$. Then, the augmented Lagrangian (2.18) reads

$$\mathcal{L}^\rho(x, z, \lambda) = \sum_{i\in\mathcal{R}} \tilde{f}_i(x_i) + \iota_C(z) + \lambda_i^\top A_i(x_i - z_i) + \frac{\rho}{2}\|A_i(x_i - z_i)\|_2^2, \quad (2.20)$$

where $\lambda^\top = (\lambda_1^\top, \dots, \lambda_N^\top)$. Now it becomes clear why introducing additional auxiliary variables and alternating minimization helps: if one fixes z in (2.20) and minimize with respect to x, the minimizations with respect to x_i are independent of each other and can be executed in parallel. This yields a parallel version of ADMM for problem (2.13a) shown in Algorithm 4. Note that the

Algorithm 4. Parallel ADMM for problem (2.13)

Initialization: z_i^0, λ_i^0 for all $i \in \mathcal{R}$, ρ

Repeat:

1) $x_i^{k+1} = \underset{x_i\in X_i}{\operatorname{argmin}}\ f_i(x_i) + \lambda_i^{k\top} A_i x_i + \frac{\rho}{2}\|A_i(x_i - z_i^k)\|_2^2, \quad i \in \mathcal{R}$ (parallel)

2) $z^{k+1} = \underset{z\in C}{\operatorname{argmin}}\ \sum_{i\in\mathcal{R}} -\lambda_i^{k\top} A_i z_i + \frac{\rho}{2}\|A_i(x_i^{k+1} - z_i)\|_2^2$ (centralized)

3) $\lambda_i^{k+1} = \lambda_i^k + \rho A_i(x_i^{k+1} - z_i^{k+1}), \quad i \in \mathcal{R}$ (parallel)

"coordination step" minimizing $\mathcal{L}^\rho$ with respect to z is a convex QP which requires central computation. However, this step can be simplified to a simple averaging step between subsystems under certain conditions [Boy+11]. This renders ADMM a *decentralized* algorithm which can be executed purely based on neighbor-to-neighbor communication. On the other hand, the multiplier update step is also executable in parallel. Note that there are further specialized variants of ADMM mostly differing in the underlying problem structures. Many of them can be reformulated in two-block form, we provide several examples in Appendix A.2.

Due to the augmentation term, if either $\mathcal{X}$ is bounded or A has full row rank, ADMM is guaranteed to converge also for non-strictly convex problems [BT89, Prop 4.2] [Boy+11, App A]. The convergence rate, however, depends on strict convexity of $\tilde{f}$ and does *not* directly carry over from the method of

multipliers.[7] If $\tilde{f}$ is convex, convergence of ADMM is generally sublinear [HY15]. Under certain conditions such as strong convexity [DY16; GB16; Shi+14] or a combination of other conditions including strict convexity [HL17], linear convergence can be achieved.[8]

Remark 4 (ADMM variants) There are various variants and derivations of ADMM differing in the order of the updates as well as in the problem setup. Moreover, multi-block variants of ADMM exist, where more than two blocks of variables are considered [HL17; LMZ15]. The overview paper [Boy+11] provides an excellent overview on the most common form of ADMM.

To illustrate that distributed optimization methods can be very useful in computer science and especially in machine learning, we present an application example of ADMM applied to support-vector machines used for classification problems.

Example 3 (ADMM for machine learning: support-vector machines) A standard method for classification are Support-Vector Machines (SVMs) aiming at separating sets of points by a hyperplane. For SVMs, distributed optimization might be interesting primarily in case of very large data sets. One reason is computational speedup, but also data privacy is a reasons why one would like to compute on multiple machines in case of sensitive data [FCG10]. An linear SVM (for two classes) can be cast as the optimization problem

$$\min_{w,b} \sum_{j \in \mathcal{Y}} \left(1 - \sigma_j (w^\top y_j - b)\right)_+ + \frac{\delta}{2} \|w\|_2^2, \tag{2.21}$$

where $(x)_+ := \max(0, x)$, $y_j \in \mathbb{R}^{n_y}$ are given data points collected in the set $\mathcal{Y} = \{1, \dots, N_y\}$ [CL11]. These points are classified into two groups labeled with $\sigma_j \in \{-1, 1\}$. The optimization variables $w \in \mathbb{R}^2$ and $b \in \mathbb{R}$ define the affine subspace separating the two groups of points.

[7] One explanation for this is the fact that the convergence rate estimates of the method of multipliers depends on the *exact* minimization of $\mathcal{L}^\rho$ jointly with respect to x *and* z. In ADMM, however, one executes a form of block-coordinate descent yielding ane inexact minimization.

[8] Moreover, ADMM is convergent for special classes of non-convex problems, cf. [WYZ19; HLR16]. However, for general non-convex problems, there is—to the best of the author's knowledge—no convergence guarantee for convergence.

We would like to distribute computation among $\mathcal{R} = \{1, \dots, R\}$ processors. For doing so, we partition the data points y_i into R groups $\mathcal{Y}_i$ with $\bigcup_{i=1,\dots,N} \mathcal{Y}_i = \mathcal{Y}$. Hence, problem (2.21) can be written as

$$\begin{aligned} \min_{\{w_i\}_{i\in\mathcal{R}},\{b_i\}_{i\in\mathcal{R}},\bar{a},\bar{b}} & \sum_{i\in\mathcal{R}} \sum_{j\in\mathcal{Y}_i} \left(1 - \sigma_j(w_i^\top y_j - b_i)\right)_+ + \frac{\delta}{2}\|w_i\|_2^2, \\ \text{subject to} \qquad & w_i = \bar{w}, \quad b_i = \bar{b}, \qquad i = 1, \dots, R \end{aligned}$$

which is in form of (2.12).[9] The augmented Lagrangian then reads

$$\begin{aligned} \mathcal{L}^\rho(\{w_i\}_{i\in\mathcal{R}},\{b_i\}_{i\in\mathcal{R}}, \bar{w}, \bar{b}) = & \sum_{i\in\mathcal{R}} \sum_{j\in\mathcal{Y}_i} \left(1 - \sigma_j(w_i^\top y_j - b_i)\right)_+ + \frac{\delta}{2}\|w_i\|_2^2 \\ & + \lambda_{wi}^\top(w_i - \bar{w}) + \lambda_{bi}(b_i - \bar{b}) + \frac{\rho}{2}\|w_i - \bar{w}\|_2^2 + \frac{\rho}{2}\|b_i - \bar{b}\|_2^2. \end{aligned}$$

ADMM leads to the iterates

$$\begin{aligned} (w_i^{k+1\top}, b_i^{k+1}) = & \arg\min_{w_i, b_i} \sum_{j\in\mathcal{Y}_i} \left(1 - \sigma_j(w_i^\top y_j - b_i)\right)_+ + \frac{\delta}{2}\|w_i\|_2^2 + \lambda_{wi}^\top(w_i - \bar{w}) \\ & + \lambda_{bi}(b_i - \bar{b}) + \frac{\rho}{2}\left(\|w_i - \bar{w}\|_2^2 + \|b_i - \bar{b}\|_2^2\right), \quad i \in \mathcal{R}. \\ (\bar{w}^{k+1\top}, \bar{b}^{k+1}) = & \frac{1}{|\mathcal{R}|} \sum_{i\in\mathcal{R}} (w_i^{k+1\top}, b_i^{k+1}), \\ (\lambda_{wi}^{k+1\top}, \lambda_{bi}^{k+1}) = & (\lambda_{wi}^{k\top}, \lambda_{bi}^k) + \rho\left((\bar{w}_i^{k+1\top}, \bar{b}_i^{k+1}) - (w_i^{k+1\top}, b_i^{k+1})\right), \qquad i \in \mathcal{R}. \end{aligned}$$

Note that the first (and by far most expensive) step of the above iterates can be implemented in parallel.[10] Note that this first step essentially solves an independent SVM problem only considering the data only from $\mathcal{Y}_i$ with two additional summands in the objective function repsresenting information form the other data sets by the global averages $\bar{w}$ and $\bar{b}$.

[9] Except for f_i being non-smooth. However, ADMM can in general also be applied to non-smooth problems as the one considered here [Boy+11].

[10] Note that the summands involving the multipliers λ_{wi} and λ_{bi} in the second step cancel out as it can be shown they are zero after the first iteration [Boy+11, Chap 7]. Moreover, observe that the parallel step can be implemented by a QP solver.

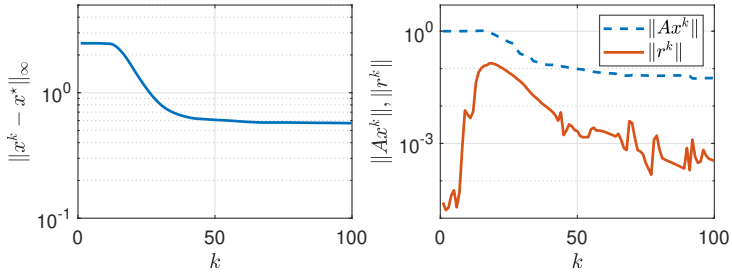


Figure 2.2: Convergence of the ADMM-based SVM.

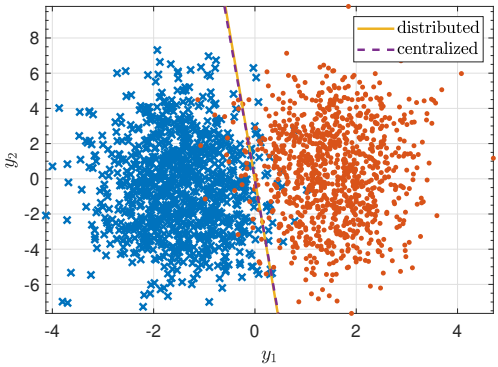


Figure 2.3: Centralized and distributed SVM result.

This implementation is especially useful for large datasets: in this case, the data can be distributed equally among all processors by choosing the same carnality for each $\mathcal{Y}_i$. This way, each subsystem needs approximately the same time to solve its local SVM. By doing so, one can (theoretically) keep the execution time constant when increasing the number of processors simultaneously with the increase in the number of data points. However, keep in mind that such decomposition makes sense only for large problems and when using multiple processors, or in case privacy is of concern. For small datasets standard centralized convex QP algorithms (e.g. interior point methods) will usually outperform ADMM.

To test the ADMM-based SVM, we consider a set of 2,000 data points distributed equally among $R = 10$ processors, where $r^k = \rho((\bar{w}_i^{k+1\top}, \bar{b}_i^{k+1}) - (\bar{w}_i^{k\top}, \bar{b}_i^k))$ is the *dual residual*, i.e. the degree of stationarity in the KKT conditions (2.2a). We use $\rho = \delta = 1$ as ADMM and SVM parameters, cf. [Boy+11, Chap 3.3]. The datasets are partitioned in the worst-possible way to exclude good solutions by chance: we choose 5 subsystems with all positive datapoints and 5 subsystems with all negative datapoints. Figure 2.2 shows the convergence of the ADMM-based SVM. One can see that ADMM converges quite slowly and only to a limited accuracy. However, Figure 2.2 depicts the resulting hyperplane computed by the centralized and the distributed SVM. The two hyperplanes are almost indistinguishable. This is typical for ADMM: it converges to medium accuracy in a hand-full iterations and then slows down. However, as for many applications this level of accuracy is sufficient—especially in context of machine learning—ADMM seems to be an excellent choice for these applications.

2.2.3 Basic ALADIN

A fundamental limitation of dual decomposition and ADMM are their convergence guarantees only for convex or strictly convex problems. One of the very few algorithms for *non-convex* problems is the Augemented Lagrangian Alternating Direction Inexact Newton (ALADIN) algorithm [HFD16]. A main advantage of ALADIN is its local quadratic convergence rate and local convergence guarantees also for non-convex problems.

In contrast to ADMM, ALADIN does *not* introduce auxiliary variables z into the problem formulation. It directly deals with the partial augmented Lagrangian of (2.12) with respect to (2.12d)

$$\mathcal{L}^\rho(x, \lambda) = \sum_{i \in \mathcal{R}} f_i(x_i) + \iota_{\mathcal{X}_i} + \lambda^\top \left(\sum_{i \in \mathcal{R}} A_i x_i - b \right) + \frac{\rho}{2} \left\| \sum_{i \in \mathcal{R}} A_i x_i - b \right\|_2^2 . \quad (2.22)$$

Consider the method of multipliers, but instead of applying a full minimization of $\mathcal{L}^\rho$ with respect to x we apply only one equality-constrained SQP step yielding

$$\begin{aligned}\min_x \sum_{i\in\mathcal{R}} \frac{1}{2}\Delta x_i^\top B_i^k \Delta x_i + \nabla f_i^\top(x_i)\Delta x_i + \lambda^{k\top}\left(\sum_{i\in\mathcal{R}} A_i(x_i+\Delta x_i) - b\right) \\ + \frac{\rho}{2}\left\|\sum_{i\in\mathcal{R}} A_i(x_i+\Delta x_i) - b\right\|_2^2 \qquad (2.23)\\ \text{subject to } \tilde{g}_i(x_i^k) + \nabla\tilde{g}_i(x_i^k)\Delta x_i = 0, \qquad \forall i\in\mathcal{R},\end{aligned}$$

where we combine equality constraints and active inequality constraints in $\tilde{g}_i(x_i)^\top = \left(g_i(x_i)^\top,\ (h_i(x_i)^\top)_{\mathcal{A}(x_i)}\right)$. Here, the matrix B_i^k is a positive definite approximation of the Hessian of the full Lagrangian, i.e. $B_i^k \approx \nabla^2_{x_i x_i}\left((f_i(x_i^k) + \gamma_i^\top g_i(x_i^k) + \mu_i^\top h_i(x_i^k)\right)$. For the multiplier update we apply the standard dual ascent step from the method of multipliers (2.11)

$$\lambda^{k+1} = \lambda^k + \alpha^k\left(\sum_{i\in\mathcal{R}} A_i x_i^{k+1} - b\right).$$

In principle, one could apply this algorithm now to equality-constrained problems. This would yield a very effective algorithm since if ρ is large enough, the QP (2.23) becomes strongly convex and thus it can be replaced by solving the KKT conditions (2.2) which is a linear system of equations. However, if inequality constraints are present, the question arises how to obtain the active set $\mathcal{A}(x^k)$. An alternative is to consider inequality constraints in (2.23), but this would make (2.23) substantially more difficult to solve, since the KKT conditions (2.2) also entail inequality constraints in this case.

ALADIN uses a different approach: it introduces a local NLP step very similar to step 1) of ADMM (Algorithm 4). This step reads

$$\min_{x_i} \sum_{i\in\mathcal{R}} \tilde{f}_i(x_i) + \lambda^{k\top} A_i x_i + \frac{\nu}{2}\|x_i - z_i^k\|_{\Sigma_i}^2, \qquad (2.24)$$

where Σ_i is a (usually diagonal) positive definite scaling matrix and where we introduce auxiliary variables z^k serving as a second iterate in ALADIN.

As an active set for (2.23), one can use the active set from the minimization of problem (2.24). This has additionally the charm that, if derivative-based numerical solvers are used to solve (2.24), one can reuse these derivatives for setting up (2.23). Note that (2.24) is separable, i.e. the minimization can be parallelized. Combining the above yields the ALADIN algorithm summarized in Algorithm 5. Note that $\tilde{g}_i(x_i^k) = 0$ in Algorithm 5 as feasibility is ensured in step 1) and that $\nabla\tilde{g}$ is the block-diagonal concatenation of all ∇g_i.

Algorithm 5. Basic ALADIN

Initialization: $z_i^0, \lambda^0, \Sigma_i \succ 0$ for all $i \in \mathcal{R}$, ν, ρ
Repeat:

1) Solve for all $i \in \mathcal{R}$

$$x_i^k = \underset{x_i \in X_i}{\operatorname{argmin}}\ f_i(x_i) + \lambda^{k\top} A_i x_i + \frac{\nu}{2}\|x_i - z_i^k\|_\Sigma^2, \qquad \text{(parallel)}$$

2) Compute $\nabla f_i(x_i^k)$, $B_i^k \approx \nabla^2_{x_i x_i}\left(f_i(x_i^k) + \gamma_i^\top g_i(x_i^k) + \mu_i^\top h_i(x_i^k)\right)$, $\nabla\tilde{g}_i(x_i^k)$.

3) Solve the coordination QP

$$\begin{aligned}\Delta x^k = \underset{\Delta x}{\operatorname{argmin}}\ & \textstyle\sum_{i\in\mathcal{R}} \frac{1}{2}\Delta x_i^\top B_i^k \Delta x_i + \nabla f_i^\top(x_i^k)\Delta x_i \qquad \text{(centralized)}\\ & + \lambda^{k\top}\left(\textstyle\sum_{i\in\mathcal{R}} A_i(x_i^k + \Delta x_i) - b\right) + \frac{\rho}{2}\|\textstyle\sum_{i\in\mathcal{R}} A_i(x_i^k + \Delta x_i) - b\|_2^2\\ & \text{subject to}\ \ \nabla\tilde{g}(x^k)\Delta x = 0.\end{aligned}$$

4) Set $z_i^{k+1} = x_i^k + \Delta x_i^k$ and $\lambda^{k+1} = \lambda^k + \rho\left(\sum_{i\in\mathcal{R}} A_i x_i - b\right)$. (parallel)

Remark 5 (Slack variables for numerical stability) In [HFD16], the QP step (2.23) is replaced by the equivalent formulation

$$\begin{aligned}&\min_{\Delta x, d} \sum_{i\in\mathcal{R}} \frac{1}{2}\Delta x_i^\top B_i^k \Delta x_i + \nabla f_i^\top(x_i^k)\Delta x_i + \lambda^{k\top} d + \frac{\mu}{2}\|d\|_2^2\\ \text{subject to}\quad & \tilde{g}_i(x_i^k) + \nabla\tilde{g}_i(x_i^k)\Delta x_i = 0 \qquad \forall i \in \mathcal{R},\\ & \sum_{i\in\mathcal{R}} A_i(x_i^k + \Delta x_i) - b = d \qquad \mid \lambda^{\mathrm{QP}},\end{aligned} \tag{2.25}$$

where $d \in \mathbb{R}^{n_c}$ are slack variables. These slack variables are important for numerical stability of ALADIN in implementations. For the understanding of ALADIN, this formulation does not add much, but due to their practical importance we will consider this formulation in the following.

Remark 6 (Computation of λ^{k+1}) Note that λ^{k+1} coincides with λ^{QP} from (2.25). This can be seen from the optimality conditions to (2.25), $\lambda^{\mathrm{QP}} = \lambda^k + \rho d = \lambda^k + \rho \left(\sum_{i \in \mathcal{R}} A_i x_i - b\right)$.

Remark 7 (Globalization of ALADIN) Note that for global convergence, i.e. convergence from remote starting points, a globalization rountine is necessary. This means that step 3) of ALADIN is replaced by

$$z_i^{k+1} = z_i^k + \alpha_1 (x_i^k - z_i^k) + \alpha_2 \Delta x_i^k, \quad \lambda^{k+1} = \lambda^k + \alpha_3 (\lambda^{\mathrm{QP}} - \lambda^k),$$

where the step-sizes α_1, α_2 and α_3 are determined by these routines. The setting $\alpha_1 = \alpha_2 = \alpha_3 = 1$ recovers step 3) of Algorithm 5, which is called the full step variant of ALADIN. One globalization routine is given in [HFD16], however it is highly centralized and computationally intense. In the present work we focus on local convergence, i.e. on convergence for initial guesses close enough to a local minimizer where a globalization routine can be omitted.

2.3 What's wrong with constraints?

In this subsection we will briefly comment on difficulties, which can arise in the application of distributed optimization algorithms to *constrained* problems. Constraints are important in many cases—particularly for applications in power systems and control which we have in mind.

2.3.1 Alternating projections in ADMM and ALADIN

Constrained problems might lead to difficulties in alternating direction methods such as ADMM. As outlined before, the core idea of ADMM is to minimize the augmented Lagrangian $\mathcal{L}^\rho$ with respect to two distinct variable blocks x and z in an *alternating fashion* to obtain separability. The name "alternating direction"

already suggests, ADMM is a kind of alternating projection method between the two types of constraints—local constraints and consensus constraints. This can be seen by the two different constraint sets $\mathcal{X}_i$ and C in the minimization steps of Algorithm 4. These alternating projections might lead to very slow convergence of ADMM as we will see next.

An example

Let us consider a convex feasibility problem (i.e. $f \equiv 0$) with a local constraint set $\mathcal{X}$

$$\min_x \iota_{\mathcal{X}}(x) \quad \text{subject to} \quad Ax = b. \tag{2.26}$$

Then ADMM (Algorithm 3 simplifies to the following alternating projection procedure

$$\begin{aligned} x^{k+1} &= \Pi_{\mathcal{X}}(z^k - u^k) \\ z^{k+1} &= \Pi_C(x^{k+1} + u^k) \\ u^{k+1} &= u^k + x^{k+1} - z^{k+1}, \end{aligned}$$

where $\Pi_C : \mathbb{R}^{n_x} \to C$ denotes the orthogonal projection on the "consensus constraint set $C = \{x \in \mathbb{R}^{n_x} \mid Ax = b\}$, see [Boy+11, Sec 3.1.1]. Let us consider the 2-dimensional example

$$C = \{x \in \mathbb{R}^2 \mid x_2 = 0\} \quad \text{and} \quad \mathcal{X} = \{x \in \mathbb{R}^2 \mid \|x - \bar{x}\|_2^2 \leq r^2\},$$

with $\bar{x} = \begin{pmatrix} 10 & 50.99 \end{pmatrix}^\top$ and $r = 51$. The set of optimal solutions is given by $\mathcal{X}^\star = \mathcal{X} \cap C = \{x \in \mathbb{R}^2 \mid (c \;\; 0)^\top, c \in [8.99 \;\; 11.01]\}$.

Figure 2.4 depicts the two feasible sets $\mathcal{X}$ and C considered in step 1) and step 2) of ADMM with the corresponding iterates starting from the initial point $z^0 = (0, \; 0.5)^\top$ and $u^0 = (0, \; 0)^\top$. One can clearly see the alternating projections in both algorithms: the iterates $\{x^k\}$ stay feasible in $\mathcal{X}$ and the iterates $\{z^k\}$ stay feasible in C—thus "alternating" between these two sets.[11]

[11] For ALADIN, the iterates $\{z^k\}$ slightly deviate from C as the consensus constraints (2.12d) are considered in the penalization term of $\mathcal{L}^\rho$.

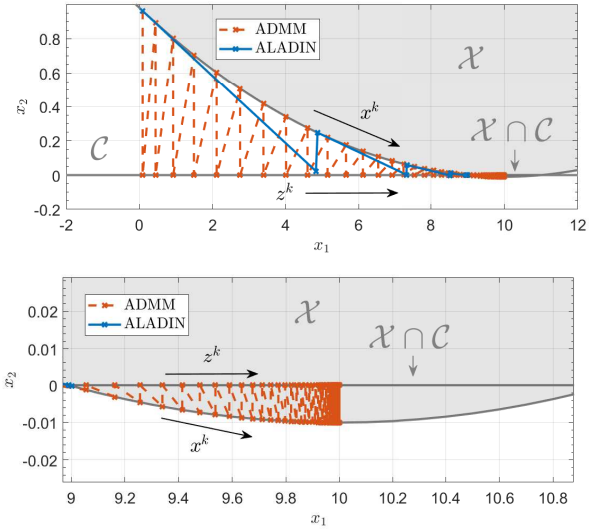


Figure 2.4: ALADIN and ADMM iterates for problem (2.26).

For ADMM convergence is slow since pure projection between both sets makes almost no progress toward their intersection. For ALADIN, the situation is different since here also curvature information of the constraint set $\mathcal{X}$ (i.e. the constraint linearizations $\nabla \tilde{g}(x^k)$ in step 2) of Algorithm 5) is considered driving the iterates much faster to $\mathcal{X} \cap C$. This can be seen from the step directions $(x^k - z^k)$ being almost tangential to the boundary of $\mathcal{X}$.

Remark 8 (Description of feasible sets) Note that considering constraint linearizations in ALADIN comes at the cost of requiring the feasible set $\mathcal{X}$ to be described by differentiable constraint functions g and h, i.e. $\mathcal{X} = \{x \in \mathbb{R}^{n_x} \mid g(x) = 0,\ h(x) \leq 0\}$ as in (2.1b). This can be seen as a restriction to the class of distributed problems, to which ALADIN is applicable. ADMM on the other hand does not have such a restriction and is thus able to handle a general convex constraint set. In view of the fact that the majority of practical applications *has* constraint sets described by differentiable functions, not considering this information means to accept a unnecessary poor performance.

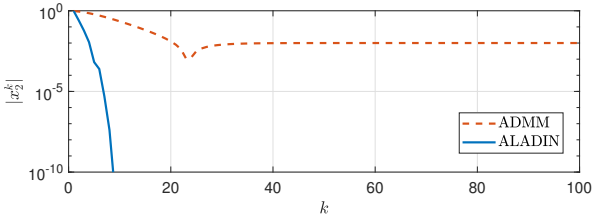


Figure 2.5: Infeasibility in C over the iteration index.

2.3.2 Integrator wind-up in ADMM

Apart from slow convergence from alternating projections, there seems to be a second effect in ADMM slowing down its performance. This effect is related to the multiplier update in step 3) of ADMM (Algorithm 4) acting as a integrator.

Figure 2.5 shows the infeasibility of the iterates $\{x^k\}$ in terms of the constraint set C, i.e. $|x_2^k|$ over the iteration index k. Here one can see that not only ALADIN converges much faster than ADMM in terms of total iterations, but also that the accuracy in the constraint violation one can reach with ADMM seems to be limited to a level of around 10^{-2}. This is surprising, as ADMM is shown to converge monotonically [HY15].

Figure 2.6 also shows the the infeasibility of the iterates $\{x^k\}$ in terms of the constraint set C but for a much larger number of iterations and additionally also the scaled dual variables u^k. Here, one can observe a very surprising effect: whereas in the primal space, there is no visible progress for more than 600 iterations, the dual variables change slightly. After about 700 iterations, the primal variables converge suddenly and to a very high accuracy.

By having a look at the dual variables, one can see why: they integrate to a value of about 6 in the first few iterations and then converge only very slowly to their optimal value of zero, which seems to hinder the primal values from convergence.

One might ask why this is the case. From Figure 2.4 one can see that ADMM performs quite large alternating projection steps, which drive the dual variable to a high value in the first iterations, since the multiplier update rule "integrates" the distance $(x^{k+1} - z^{k+1})$. Once the z^k iterates reached the feasible set, the

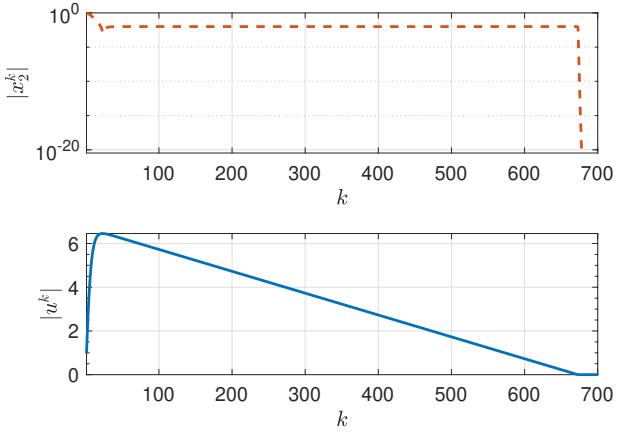


Figure 2.6: Infeasibility in C and scaled dual variables.

alternating behavior continues although the iterates z^k are already feasible. As in this new area of oscillation (Figure 2.4, second plot), the distance $(x^{k+1} - z^{k+1})$ is quite small, it takes several hundreds of iterations to dismount the before integrated value of the multipliers. Once they reached their optimal value, ADMM suddenly converges. Observe that for this example, the behavior is independent of ρ, since ρ does not appear in the ADMM iterations. Thus, this effect can *not* be overcome by tuning.

To the best of our knowledge, this effect is not investigated in the literature in detail so far. Moreover, it seems to be one reason for very slow convergence of ADMM for some OPF problems (cf. Section 5.4.1).

2.3.3 Non-convex constraints

Many distributed algorithms require the feasible set to be convex. If one nonetheless applies these algorithms to problems with non-convex constraints, the convergence is not guaranteed. Thus, the question arises, whether there is a particular difficulty for *distributed* algorithms to handle non-convex constraints. Next, we will show that this is not the case. More specifically, we will

show an example, where *all here considered algorithms diverge—including centralized methods* such as the method of multipliers and SQP diverge contradicting the common wisdom that they can achieve convergence to a local minimizer from an *arbitrary* initialization by considering suitable globalization routines.

With this, we would like to emphasized that there is an *inherent difficulty* associated with certain kinds of non-convex constraints from which all algorithms suffer—not only the distributed ones. This should shape our expectation that it is hardly possible to design distributed algorithms which are able to handle general non-convex constraints. The best we can achieve also here is local convergence—i.e. convergence from an initial point close enough to a local minimizer.[12]

What's wrong with non-convex constraints?

Consider the non-convex problem

$$\min_{x\in\mathbb{R}^2} 10^{-2}\Big((x_1-8)^2+(x_2-1)^2\Big) \tag{2.27a}$$

$$\text{subject to} \quad x_2-\sin(x_1)=0 \tag{2.27b}$$

$$(0.15,\ -1)\,x=0. \tag{2.27c}$$

This problem can be written as

$$\min_{x} \quad 10^{-2}\left((x_1-8)^2+(x_2-1)^2\right)+i_{\mathcal{X}_1}(x)+i_{\mathcal{X}_2}(x) \tag{2.28}$$

with a non-convex constraint set $\mathcal{X}_1=\{x\in\mathbb{R}^2 \mid x_2-\sin(x_1)=0\}$ and a constraint set $\mathcal{X}_2=\{x\in\mathbb{R}^2 \mid (0.15,\ -1)\,x=0\}$. The contour lines of the objective and the feasible are depicted in Figure 2.7. The feasible set consists of three points $\mathcal{X}=\{(0,0)^\top,\pm(2.72,\ 0.41)^\top\}$, one of which ($x^\star=(2.72,\ 0.41)^\top$) is globally optimal.

[12] One expectation are centralized global optimization algorithms such as α-branch-and-bound [Ste17]. However, these global optimization techniques are typically very expensive and often due to that not applicable to large problems.

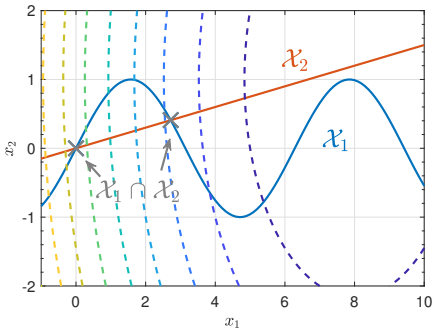


Figure 2.7: Contour plots and feasible set of problem (2.27).

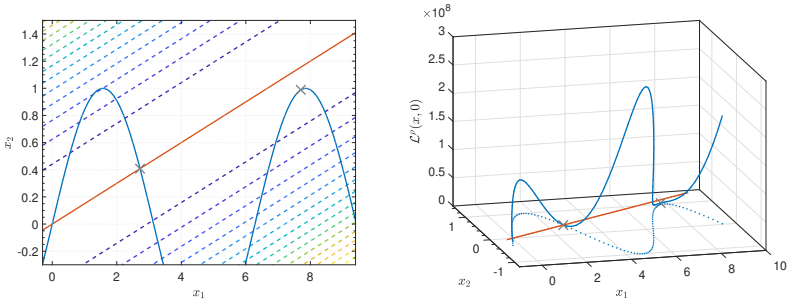
Three of the four considered algorithms—ADMM, ALADIN and the method of multipliers—are based on minimizing the augmented Lagrangian which for $\lambda = 0$ reads

$$\mathcal{L}^\rho(x, 0) = 10^{-2}\left((x_1 - 8)^2 + (x_2 - 1)^2\right) + i_{\mathcal{X}_1}(x) + \frac{\rho}{2}\left\|(0.15,\ -1)\, x\right\|_2^2.$$

The theory of augmented Lagrangian-based methods predicts that if we iteratively minimize $\mathcal{L}^\rho$ for a fixed multiplier λ^k and we let $\rho \to \infty$, then these methods will converge to the globally optimal solution to (2.27) regardless of the update rule for λ^k [Ber99, Prop. 4.2.1]. At first glance, this sounds very promising, but let us have a closer look on $\mathcal{L}^\rho$ when increasing ρ. Figure 2.8 shows the contour lines and the feasible set of the problem

$$\min_x \mathcal{L}^\rho(x, 0) \tag{2.29}$$

for $\rho = 10^8$. One can clearly see that $\mathcal{L}^\rho(x, 0)$ has two local minima in the plotted range: one at the global optimal solution to problem (2.27), $(2.72,\ 0.41)^\top$ but also another one at $(7.7,\ 0.98)^\top$ which is not even a feasible point. This highlights an inherent practical difficulty of augmented Lagrangian methods for non-convex constraints: if $\mathcal{L}^\rho$ is iteratively minimized with *local solvers* (which is often the case in practice), then the method might converge to a point



(a) Top view.

(b) Side view.

Figure 2.8: Feasible set and contours of (2.29) for $\rho = 10^8$.

which is neither feasible nor optimal.[13] The premise of the before mentioned theorem is that $\mathcal{L}^\rho$ is minimized *globally*, which can not be guaranteed here due to using local solvers. Consequently, this can be observed in ADMM, ALADIN and the method of multipliers if their subproblems are solved with local solvers.

Numerical behavior

Figure 2.9 shows the iterates for ADMM, ALADIN and the method of multipliers from an initial point $x^0 = (5, \ -0.5)$ and $\lambda^0 = 0$. One can observe the convergence to the infeasible point $(2.72, \ 0.41)^\top$ for all these methods and similar to the previous chapter the "alternating behavior" of ALADIN and ADMM being projecting between the two constraint sets defined by g and $Ax = b$.

Moreover, for SQP, after several iterations close to the infeasible point, SQP produces iterates far from any feasible point (indicated by the blue arrow in Figure 2.9). The reason for this is that ∇g and A become very close to linearly dependent at $(2.72, \ 0.41)^\top$ leading to intersection points of the spaces spanned by ∇g and A very far away from the previous iterate. Globalization routines

[13] This effect is even strengthened if the local solvers are initialized at the previous iterate which is also often done in practice for computational efficiency reasons.

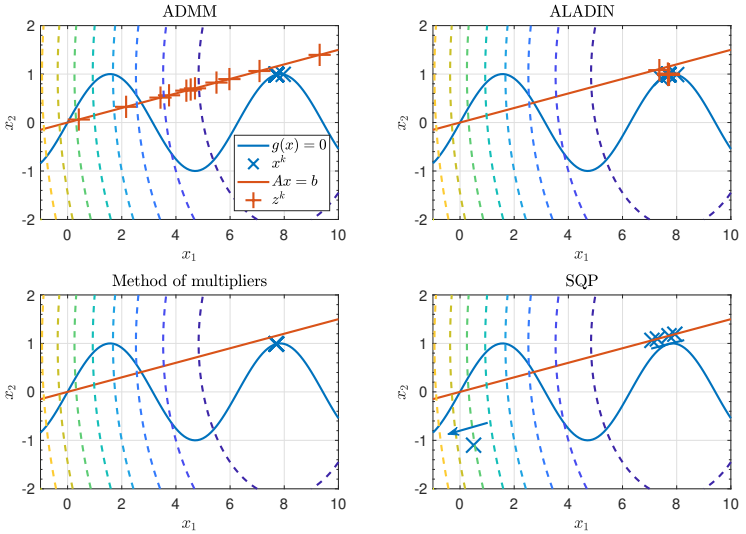


Figure 2.9: Iterates of ADMM, ALADIN, the method of multipliers, and SQP for problem (2.27).

do also not help here. Most of the global convergence proofs for SQP require either an a priori assumption boundedness of the iterates $\{x^k\}$, $\{\lambda^k\}$ [Sol09], or they require linearly independent constraint linearizations [BT95, Thm 4.1], which are both violated in this case. Even if these assumptions hold, the convergence of SQP methods is typically guaranteed to a *stationary point* of some merit function such as $\mathcal{L}^\rho$ [BT95, Thm 4.2]. However, these stationary points do not necessarily correspond to a local minimizer as we observed in our example.

Note that this effect can not occur for convex problems. In this case, all subproblems are also convex and solvers necessarily return global optima. In summary, we can conclude the following: Especially non-convex constraints are difficult to handle—not only for distributed but also for centralized algorithms. Thus, designing distributed algorithms with convergence guarantees from *arbitrary* initial points might be out of reach. However, in practical problems we often have initial guesses close enough to local minimizers making

these algorithms nonetheless very successful in many domains. A similar class of counterexamples for interior-point methods can be found in [WB00].

3 A Survey on Distributed Optimization

In this chapter, we provide an overview on the state-of-the art of distributed optimization algorithms. Moreover, we investigate what makes these algorithms hard to apply in many practical applications. The amount of literature on distributed optimization grows very fast and this overview is far from being complete. However, we tried to identify main lines of research and aimed at identifying important works for each of these lines.

We categorize the research on distributed optimization along three main lines of research:

- primal-dual algorithms,
- primal algorithms,
- and internal decomposition algorithms.

Whereas primal-dual methods are mainly based on Lagrangian relaxation (cf. Section 2.2), primal methods work purely in primal space, i.e. they do typically *not* update any form of Lagrange multipliers. The third line of research considers internal decomposition methods. These methods decompose operations of standard nonlinear programming algorithms such as solving a linear system of equations. We begin with primal-dual methods as one of the earliest approaches on distributed optimization.

3.1 Primal-dual algorithms

Early works on distributed optimization trace back to Everett Dantzig, Wolfe and Benders [Eve63; DW60; Ben62] in the 1960s. These first works mainly

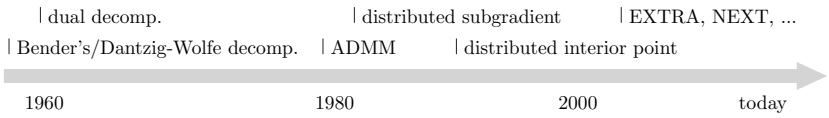


Figure 3.1: Timeline of distributed optimization algorithms.

considered Lagrangian relaxation for strictly convex problems and decomposition methods for linear programs. Later, the Lagrangian relaxation was combined with augmented Lagrangian techniques developed mainly by Hestenes, Powell and Miele [Pow69; Hes69; Mie+71; Mie+72] to improve numerical stability and to provide guarantees also for convex but not strictly convex problems. This led to first versions of ADMM with improved convergence guarantees and improved practical convergence [GM76; GM75; EB92]. Many of these duality-based works are summarized excellent textbooks by Bertsekas and Tsitsiklis [BT89] and Censior and Zenios [CZ97]. A timeline of distributed optimization approaches is shown in Figure 3.1.

Renewed interest in the late 2000s

Distributed optimization gained new interest in the late 2000s mainly in the field of machine learning and imaging science, where ADMM outperformed state-of-the art methods in certain applications [ABF10; SMG10; GM12]. Moreover, new applications in signal recovery emerged [CP07; CP11]. The main motivation here was computational speedup, i.e. to find methods for parallel computing.[1] Moreover, state-of-the-art methods have been shown to be a special case of ADMM [GO09; Gol+14] allowing their treatment in a unified framework.

Duality-based optimization methods were used in communication networks [Chi+07] beginning already in the late 1990s [KMT98; LL99]. Similar approaches were used in wireless sensor networks [MBG10; SRG08], in signal

[1] A nice example for this need is given in [BMN01]. There, the authors consider a image-reconstruction problem from from positron emission tomography. Even an evaluation of one gradient of the objective took 15 to 45 min there, so there is no chance for using Newton-based schemes such as IP or SQP methods.

processing [ZGC09] and in certain fields of machine learning (support-vector machines) [FCG10]. A different version of dual decomposition is established in [NS08] called *proximal center method*. Herein—instead of minimizing the augmented Lagrangian in an alternating fashion to achieve separability—the author adds two linear proximal terms which are separable and lead to a differentiable dual function. An early dual-decomposition based approach for a multi-agent setting was presented in [TTM11]. The highly influential paper [Boy+11] showed that many of the above works can be treated in a unified framework based on ADMM. The importance of this framework lies in its generality—the convergence analysis of [Boy+11] carries over to the above applications. An even more general convergence analysis of decomposition methods (including ADMM) based on augmented Lagrangians is given in [ST14]. A dual method combining with smoothing and an excessive gap technique for convex problems is presented in [TDSD13]. The numerical results in this work are promising and the method has a main advantage over other primal-dual methods: the parameters are updated automatically. Using an accelerated scheme in the dual step yields a convergence rate of $O(1/k^2)$ for general convex problems. A generalization is given in [TND16] including techniques from [Nes05a; Nes05b] allowing for inexact solutions of the subproblems and providing automatic parameter updates.

The works starting with [WO12] consider a network setting, i.e. they introduce consensus constraints on characteristic matrices of the underlying communication graph (incidence matrix, Laplace matrix, cf. Appendix A.2). [Shi+14] considers a problem formulation in form of (A.10). This leads to an *edge-based* formulation (ADMM maintains two Lagrange multipliers per edge). [MO17] develops a *node-based* formulation, where one Lagrange multiplier vector is maintained for each node in the network, where the network constraint is encoded based on a Laplacian formulation similar to (A.12). Both cases lead to sublinear/linear convergence under convexity/strong convexity assumptions. The very recent work [Mao+20] presents a decentralized method for unconstrained non-convex consensus optimization with guarantees in a network setting.

Recent works considering non-convex objectives

Recently, primal-dual algorithms were successfully extended to problems with non-convex objective function, but convex constraint set with guarantees. An ADMM-like algorithm, where the augmented Lagrangian is minimized in a block-coordinate descent scheme, is presented in [HJ14] based on the convergence analysis of [ABS13]. The analysis is based on new concepts beyond convexity such as the Kurdyka-Łojasiewicz (KL) property and Lipschitz/regularity assumptions. The KL property is for example fulfilled, if the objective and the constraint functions are-semi algebraic. This line of analysis is combined in [HJ16] with generalized equations from [ZA10] to a distributed control setting. The KL framework is developed further in [BST18; ST19a]. Approaches applying ADMM in the field of optimal control can be found in [KCD15; RCG17; BG19]. A dual second order method decomposing strictly convex QPs from optimal control along the time-axis is presented in [FSD15].

The works [Cha+16b; Cha+16a] show that ADMM with standard consensus and strongly convex f_is converges linearly even under asynchronous operation with bounded communication delay. [CDZ15] and [CZ17] present an ADMM-flavoured method providing convergence guarantees also for non-convex f_i under strong second-order sufficient conditions but with convex constraint sets. Moreover, [HLR16; LP15] analyze the convergence properties of ADMM for non-convex problems including certain classes of non-convex constraints. An more general convergence analysis is given in [WYZ19] including the [HLR16; LP15] as special cases. The line of analysis of [WYZ19] is extended in [GGC20] to problems, where the consensus constraint (2.12d) can be a multiaffine mapping.

Remark 9 (Relation to operator splitting methods) There are excellent works generalizing primal-dual distributed algorithms in the framework of *operator splitting* and *proximal* methods [CP11; Sta+16; PB14]. This perspective paves the way for unified convergence analysis. Moreover, it leads to suggestions for effective preconditioning [GB15; GB17]. However, this point of view is beyond the scope of the present work as it requires a large amount additional mathematical prerequisites from operator splitting theory. We refer to [BC11] and the above works for further details.

Table 3.1: Typical properties of primal-dual decentralized algorithms.

update	idea	convergence	pros/cons
$x^{k+1} \in \operatorname{argmin}_x \mathcal{L}^\rho(x, \lambda^k)$[2] $\lambda^{k+1} = \lambda^k + \rho(Ax - b)$ $\mathcal{L}^\rho$: augmented Lag. ρ : stepsize	maximize dual + evaluate primal distributedly	$f \in \mathcal{S} : \mathcal{O}(1/k)$ / $\mathcal{O}(1/k^2)$ $f \in \mathcal{S}_\mu : \mathcal{O}(\tau^k)$	+ decentralized + robust + extension to certain non-convex settings - tuning - non-conv constraints?

Pros and cons of primal-dual methods

Primal-dual methods are very successfully applied in many contexts—especially for convex problems. For certain applications—e.g. in embedded optimal control, signal recovery and machine learning—they lead to promising performance [Ste+20; CP07]. However, most of the primal-dual approaches suffer from two drawbacks:

- the constraint-set is often restricted to be convex;
- and the convergence rate is typically linear/sublinear for strongly convex/convex problems.

We refer to [DY16; GB16; PSB14] and [HY12] for a detailed convergence analysis.

Recent overviews on decentralized optimization considering primal and dual approaches can be found in [NOR18; NOW18; Boy+11; Yan+19; HM11]. Examples on the application of ADMM to machine learning problems can be found in [Tay+16; Sul+19]. Recent surveys on optimization methods for machine learning including primal and dual methods can be found in [Ber15b; BCN18; SNW12]. Overviews on splitting methods for control with emphasis on time-wise decomposition are given in [Sta+16; Fer+17].

Typical properties of primal-dual algorithms are summarized in Table 3.1.

[2] By setting $\rho = 0$ one obtains dual decomposition.

3.2 Primal algorithms

The second main line of reserach are primal first-order methods. These methods are usually decentralized variants of the gradient/subgradient or proximal gradient method producing iterates purely *in the primal space*. Hence, they do usually not maintain any form of Lagrange multipliers. We give a brief overview on latest developments in a network setting. For a recent and comprehensive analysis on their centralized counterparts we refer to [Bec17; Ber15a].

Classical approaches

In contrast to dual methods, primal distributed methods are distributed *in the design* of the optimization algorithms. One of the most general settings is given in [TBA86], which is also one of the earliest and most influential papers on primal methods in distributed optimization. To illustrate the idea of primal methods and to be able to classify relevant literature, we give a simplified version of the iteration scheme typically used in primal methods (assuming deterministics, synchronization, no communication delays and a one-dimensional decision vector). This (simplified) iteration reads

$$x^{k+1} = W^k x^k - \Lambda^k s^k. \tag{3.1}$$

Here, W^k is a so-called *mixing matrix* and s^k encodes some form of gradient/subgradient or Newton step with respect to the individual objective function terms f_i, and Λ^k is a diagonal matrix providing positive stepsizes for each s_i^k. The matrix $W^k x^k$ can be seen as a kind of averaging step between neighbors driving the individual x_i toward a consensus and the vector s^k contains some desirable direction of descent in the objective function driving the iteration towards optimality. The connectivity of the Network is here encoded in the matrix W^k (which might be time-varying).[3] Convergence of these methods is guaranteed in a very general setting including asynchronous operation and communication delays [Ber83; TBA86]. An example for an early application of this framework to distributed reinforcement learning can be found in [Tsi94].

[3] The matrix W^k has to fulfill certain assumptions, e.g. it has to be *stochastic* or *doubly stochastic* meaning that all entries have to be non-negative and the column sums of W^k have to be all one.

Consensus and first-generation algorithms

In the 2000s, so-called *consensus algorithms* gained significant interest [JLM03; Boy+05; OFM07; OT09; Blo+05].[4] Consensus algorithms compute (roughly speaking) weighted-averages between neighbored agents sequentially until a certain level of consensus (i.e. level of "equalness" in the primal variables) is reached. Thus, the iteration of consensus algorithms is $x^{k+1} = Ax^k$. By using tools from classical discrete-time control theory, one can accelerate convergence of consensus algorithms by choosing A in a certain manner. From this perspective it seems natural that consensus algorithms are closely related to the before-mentioned primal algorithms. The work [BT07] showed that some of them are a special case of the earlier work [TBA86], where roughly speaking speaking the vector s^k in the primal optimization framework (3.1) is zero. However, these development renewed interest in decentralized primal algorithms and led to the development of decentralized subgradient methods (DSG) for non-smooth f_is (e.g. for $l1$-regularized problems) [NO09]. Thereby the authors exploit that consensus problem can equivalently be formulated as a *feasibility problem*, i.e. one minimizes a zero function encoding consensus as a constraint which is mathematically described by a constraint set $\mathcal{X} = \{x \in \mathbb{R}^{n_x} \mid x_1 = x_2 =, \ldots, = x_R\}$ [Ned+14]. The decentralized subgradient method was extended to a stochastic setting with subgradient-projection where all agents have knowledge over a global constraint set and thus also decentralized *constrained* and *non-smooth* optimization became possible [SRNV10; Ned11].

In [DAW12], DSG was extended to a decentralized proximal gradient method based on [Nes09; DAW10] allowing for constraints in a deterministic, synchronous and undelayed setting. It also analyzes the influence of the network structure on the convergence. In the same setting, a decentralized primal-dual subgradient method was proposed in [ZM12]. All the above methods require a diminishing step site to converge to an optimal solution—one reason for making

[4] A consensus problem is a problem, where multiple entities have to agree on something. Examples for this can be opinion dynamics, where after a while the population agrees on a certain political decision; sensor networks where multiple measurements have to be aggregated, i.e. there has to be a "consent" on the measured value; formation control, where autonomously flying objects have to agree on a certain formation, or self-organize biological systems, where swarms of animals (e.g. fishes) move in a certain direction.

them slow compared to other methods [NOR18]. To accelerate convergence, primal methods based on Nesterov's accelerated gradient ([Nes83]) for unconstrained problems over networks have been proposed in [CO12; JXM14]. They exhibit an improved sublinear convergence rate. The work [YLY16] analyzes the convergence of DSG schemes in case of a fixed stepsize. Therein authors show that in case of a fixed stepsize, the accuracy of DSG is limited by the stepsize, i.e. DSG converges to a more exact solution with a smaller stepsize. On the other hand, this slows down convergence and hence there is a tradeoff between convergence speed and achievable accuracy which can be seen as a drawback of DSG methods. This effect is called the *exactness-speed dilemma.* An extension of the analysis of DSG and prox-DSG methods to the non-convex regime is given in [ZY18] showing that most properties from the convex setting are preserved. This work considers new analysis tools suitable for the non-convex case and shows convergence to stationary points (and to limit cycles) under almost the same conditions as in the convex case. An sufficient conditions for optimal convergence in time-varying networks applicable to the algorithms before is given in [Rog+19].

Second-generation algorithms

A major drawback of the algorithms from before is the need for a diminishing stepsize to guarantee their converges making them typically slow. A new type of decentralized primal method called *Exact First Order Algorithm (EXTRA)* was developed in [Shi+15b] overcoming this difficulty. The main idea there is to introduce a correction term in the decentralized gradient schemes from before, compensating the non-stationarity of the f_is in the limit. The correction term is a sum of previous iterates—hence, one has to maintain one additional vector while iterating. The update formulas are

$$x^{k+1} = W^k x^k - \Lambda^k y^k, \tag{3.2a}$$

$$y^{k+1} = W^k y^k + \nabla f(x^{k+1}) - \nabla f(x^k). \tag{3.2b}$$

Here, the first term in (3.2b), $W^k y^k$, is the newly introduced compensation term. By doing so one yields convergence also *for a fixed step size* overcoming the exactness-speed dilemma. EXTRA yields sublinear convergence for convex objective functions and linear convergence for strictly convex objec-

tive functions. An extension to constrained problems based on the proximal gradient method is given in [Shi+15a]. Also numerically, EXTRA has shown to outperform existing DSG schemes. A similar scheme (Aug-DGM) was proposed in [Xu+15] allowing for uncoordinated stepsizes. Algorithms very similar to EXTRA and Aug-DGM haven been proposed in [Ned+17; QL18]. In [Ned+17], the developed (DIGing) algorithm provides convergence guarantees for strongly convex functions over time-varying graphs using new arguments from control theory (small-gain theorem). The authors of [QL18] prove sublinear convergence for convex functions. An extension of [Ned+17] to uncoordinated stepsizes is given in [NOS17]. A Nesterov-accelerated (Acc-DNGD) is given in [QL19] with significantly improved sublinear convergence rate. Asynchronous operation and time-varying directed graphs with linear convergence proof is considered in [Pu+20] with a similar approach given in [XK18]. [MJ18] presents and algorithm similar to DIGing with improved practical convergence and a substantial reduction in communication. Non-convex objectives are considered in [TT17].

A different line of research in distributed primal methods was proposed in [Scu+14]. Therein, the main idea based on [RHL13] is to use a proximal-gradient like scheme, where non-convex parts of the objective are linearized and the evaluation of the proximal operator is done in a Jacobi or Gauss-Seidel like procedure. This leads to convergence to stationary points also in case of non-convex f with a convex constraint set. Note that this method also needs a diminishing stepsize. These ideas are combined with a dynamic consensus algorithm from [ZM10] leading to a very effective algorithm [DLS16] called NEXT with an empirically linear rate of convergence and convergence speed similar to ADMM. An extension to arbitrary and time-varying digraphs is given in [SSP16] (SONATA) containing NEXT, Aug-DGM and DIGing as special cases.

Decentralized Newton-methods

Another main line of research is primal decentralized optimization based on Newton-Methods. The article [JOZ09] considers a decentralized Newton scheme very similar to the one we will use later for bi-level ALADIN. The consensus constraint is defined via a graph incidence matrix, i.e. each node maintains only one decision variable. The scheme is unconstrained. The New-

ton step is computed via a decentralized consensus scheme. Errors in the step computation are included in the convergence analysis. A similar approach is proposed in [TBJ19], where in contrast to the method before, the dual Hessian is converted into a system of symmetric diagonally dominant (SDD) matrices and this new (lifted) system is solved by a decentralized solver specifically designed for SDD systems. A combination of Newton based ideas with average consensus is given in [MLR17; Mok+16] which converge quite slow even for strongly convex problems. The approach in [Var+16] is at least able to gain a practically linear rate of convergence.

Pros and cons of primal methods

A main advantage of primal methods are that thy often come with advanced features such as asynchronous operation, considering time-delays or a changing network topology. However, their convergence is typically very slow although second-generation algorithms made significant progress towards acceleration of convergence. Moreover, having optimization problems from power systems or for distributed optimal control in mind, the current state-of-the art primal methods have the major drawback that they do often not consider constraints. If constraints are considered, they are typically assumed to be convex. Moreover, knowledge of the global constraint set by all agents is sometimes required.

We refer to [NOW18; NOW18; Ned14; Yan+19] for excellent and more comprehensive reviews of distributed primal methods.

As constraints are of major importance in the applications from power systems and control we have in mind, we do not consider primal methods further in this thesis.

[19] The convergence of first-generation algorithm is sublinear mainly due to the required diminishing stepsize. If a constant stepsize is used, however, convergence might be faster but the "exactness" of the solution is limited in that case [NOR18; YLY16]. This effect is called the *exactness-speed dilemma*.

Table 3.2: Typical properties of first and second generation primal decentralized algorithms.

	update	idea	convergence	pros/cons
1st generation	$x^{k+1} = W^k x^k - \Lambda^k s^k$ W^k : mixing matrix Λ^k : stepsize s^k : step direction	consensus + gradient/ Newton step	$f \in \mathcal{S}, \mathcal{S}_\mu$: $O(1/\sqrt{k})$ / $O(1/k)$[19]	+ decentralized + very simple iteration + simple subproblems + (changing topology) + (directed graphs) - convex $\mathcal{X}$ required - exactness-speed dilemma[19] - very slow
2nd generation	$x^{k+1} = W^k x^k - \Lambda^k y^k$ $z^{k+1} = \nabla f(x^{k+1}) - \nabla f(x^k)$ $y^{k+1} = W^k y^k + z^{k+1}$ W^k : mixing matrix Λ^k : stepsizes	consensus + gradient tracking	$f \in \mathcal{S} : O(1/k)$ $f \in \mathcal{S}_\mu : O(\tau^k)$	+ no exactness-speed dilemma + decentralized + simple iteration + simple subproblems + (changing topology) + (directed graphs) - convex $\mathcal{X}$ required - slow

3.3 Internal decomposition methods

The third main line of research are *internal decomposition* methods. Internal decomposition aims at distributing certain steps from centralized nonlinear programming methods such as interior point methods or SQP. By doing so, the convergence guarantees of the "host algorithm" are inherited. In interior point and SQP, the most expensive centralized operation is solving a liner system of equations (KKT systems) in each iteration. Whereas for interior point methods decomposition of the KKT system is often straight-forward, for SQP methods the situation is a bit more difficult. Here, one has to consider an extra routine detecting the active set or, alternatively, use a method which can handle constraints as an inner algorithm.

Classically, parallel linear algebra routines such as the Gauss-Seidel, the Jacobi iteration or the conjugate gradient method were employed for distributed computation of the KKT system [BT89, Chap 2], [Saa03, Chap 4]. However, these classical methods usually decompose along rows/columns of the KKT matrix aiming for load balancing and they are thus typically not directly applicable to

a multi-agent setting. Domain decomposition methods (or Schur-complement methods) [Saa03, Chap 14] are more suited for the multi-agent setting and we will use similar techniques in Chapter 4.

Distributed interior-point and SQP methods

Early works on distributed interior point methods for convex problems in form of (2.12) start with [HOS01], where the KKT system is solved by specialized solvers for block-tridiagonal systems. A Schur-complement-based approach is presented in [GG07] and [GG09], where [GG09] efficiently exploits nested tree structures. The work [NS09] combines dual decomposition with ideas from interior point methods, where the KKT system is condensed based on a Schur-complement technique and subsequently solved by a centralized linear solver. A generalization thereof is presented in [Din+13] coming with a novel convergence analysis allowing for inexact solutions of the subproblems. The follow-up work [LT20] generalized this work for broader problem classes based on tools from [ST19b] and a polynomial-time complexity is established. A general structure-exploiting interior point method for problems in form of (2.12) is presented in [PHA14], where the KKT system is solved via ADMM.

Examples for distributed implementations of interior point methods of time-dependent problems (e.g. from optimal control) can be found in [ZLB08; Kan+14; Wor+14; HSS17] In these works, decomposition is typically done *over time*, i.e. the problems are often in form of (2.12) but where $\mathcal{R}$ is a *time index* rather than a set of subsystems. An augmented-Lagrangian based parallel interior point method for this setting is given in [CSL16]. One recent example of using Schur-complement techniques is implemented in the PARDISO solver [Sch+01] in combination with the very successful interior point method implementation IPOPT [WB06; Ver+17; KFS18; KKS20a; KKS20b]. The main focus of these implementations is the applications to time-dependent problems from power systems such as multi-stage optimal power flow or security-constrained optimal power flow—however the methods used there are in principle applicable to many problems with a similar sparsity structure. A similar approach with application to spatial decomposition in power systems is presented in [Lu+18].

Table 3.3: Typical properties of internal decomposition methods.

update	idea	convergence	pros/cons
$\begin{pmatrix} -g^k \\ b \end{pmatrix} = \begin{pmatrix} H^k & A^{k\top} \\ A^k & 0 \end{pmatrix} \begin{pmatrix} \Delta x^k \\ \Delta \lambda^k \end{pmatrix}$ $x^{k+1} = x^k + \alpha^k \Delta x^k$ $\lambda^{k+1} = \lambda^k + \alpha^k \Delta \lambda^k$ H^k : block-Hessian A^k : structured Jacobian	exploit structure in H, A (+ solve decentralized)	$f \in \mathcal{S}$:[5] $O(\tau^{(1/\tau)^k})$ / $O(k^{-k})$ / $O(\tau^k)$	+ very fast + non-convex constraints - steps expensive - distributed - no advanced features (changing topology etc.) - globalization

For a multi-agent setting, an approach based on primal barrier functions is presented in [BPJ17] and a primal-dual based approach is presented in [BJ17]. These works guarantee convergence with decentralized barrier parameter updates.

An early partially parallel SQP version is presented in [Lei+03]. [Sch15] presents a commercial distributed SQP method.

Pros and cons of internal decomposition methods

Internal decomposition methods are in general well-suited for constrained optimization. They typically consider some form of constraint-linearization in their step computation, thus the curvature information of the constraints is effectively exploited often leading to very fast convergence of these methods also in the constrained case. However, as all of these methods are second-order (i.e. they use Hessian and Jacobian information), step-computation is in general much more expensive. Using Schur-complement and reduced-space techniques can substantially reduce the dimension of the KKT system. However, solving the KKT system is in general highly centralized. Moreover, these methods are typically *not* tailored to a multi-agent setting. They are rather designed for parallel computing.

[5] The convergence rate depends on the precision of solving the KKT system and on the "exactness" of the Hessian H^k. Exact Hessians with an exact solutions to the KKT system leads to quadratic convergence, approximations like BFGS lead to superlinear convergence, cf. Example 2.

Table 3.4: Overview on main lines of research for distributed optimization.

	primal-dual	*primal*		*internal decomp.*
		1st generation	2nd generation	
examples	dual decomp., ADMM, ...	consensus, distributed subgradient, ...	EXTRA, NEXT, DIGing, ...	distributed IP, distributed SQP
distribution via	*dualization*, *alternating direction*	*averaging*	*averaging*	*linear algebra*, *dualization*
constraints via	projection	projection	projection	barrier/active set
section	Section 3.1	Section 3.2		Section 3.3
overviews in	[HM11; Boy+11] [Sta+16; NOR18] [NOW18; Yan+19]	[NOW18; NOW18] [Ned14; Yan+19]		[Wor+14; ZLB08]

For our applications in power systems and control, techniques from internal decomposition seem to be excellent fits as we often have non-convex objectives and constraints. However, one challenge is to reduce the complexity for solving the KKT system.

Mixed approaches

Finally, there are a few algorithms which are hard to categorize along the above lines of research. One of them is an alternating trust region method [HJ17] which converges linearly to local minimizers. The second one is the Augmented Lagrangian Alternating Direction Inexact Newton (ALADIN) method [HFD16]. ALADIN can be viewed as a mix between primal-dual and SQP methods combining distributed optimization with the fast (superlinear or even quadratic) convergence properties of SQP. In contrast to many other methods from before, these two methods are able to handle non-convex constraints. A decomposition scheme for time-wise decomposition of optimal control problems based on ALADIN is presented in [Kou+16].

3.4 Comparison of algorithms

Let us assess the before mentioned algorithm classes in view of desired properties for distributed optimization. Classical primal-dual algorithms such as ADMM and dual decomposition are well investigated and work robustly for a wide range of practical (convex) problems. Moreover, convergence guarantees have recently been extended to special classes of non-convex problems. For separable problem structures, they are able to operate in a decentralized fashion although centralized parameter tuning might be required. Their amount of communication is usually very small and recently also advanced features such as operating on directed graphs or asynchronous operation have been investigated. However, in case of constrained problems, certain undesired alternating projection effects might occur (cf. Section 2.3).

Primal algorithms do very well in decentralized operation as they do not assume a special structure of the problem and the communication graph can be chosen independently from the problem formulation. Moreover, similar to primal-dual methods, the communication overhead per iteration is quite small and many of these algorithms are able to operate on directly graphs with changing topology and asynchronously. On the other hand, at least first-generation primal algorithms are very slow. The convergence rate of second-generation algorithms can be similar to primal-dual methods. A drawback of primal algorithms is that considering constraints (even convex ones) is often not possible.

Internal decomposition methods are often not decentralized as they often aim for parallel computing and computational speedup than for a multi-agent setting. As they often solve a reduced KKT system, the communication overhead is quite large and the coordination problem might be expensive to solve due to additional centralized operations such as the update of barrier parameter in interior point methods. However, very fast convergence is often guaranteed also for problems with non-convex objective *and* constraints.

Lastly, ALADIN is a mixture between primal-dual methods and internal decomposition methods inheriting the fast convergence properties of internal decomposition methods and the convergence guarantees for constrained non-convex problems. Moreover—in contrast to internal decomposition methods—ALADIN offers of distribution by detecting the active constraints locally and

Table 3.5: Existing works in view of desired properties for distributed algorithms.

	primal-dual	primal	internal dec.	ALADIN
decentralization	+	++	- -	-
convergence guarantees	conv.	conv.	non-conv	non-conv
convergence speed	o	(- -), o	++	+++
communication	++	++	- - -	- -
broad applicability	++	- -	+	+
constraint-handling	+	–	++	++
advanced features[6]	+	++	- -	- -

reducing the coordination step to a linear system of equations. However, compared to primal-dual and primal methods, its coordination and communication effort is quite large. The properties of primal-dual, primal and internal decomposition methods in view of the requirements for distributed optimization are summarized in Table 3.5. Therein (+) and (++) indicate that the considered algorithm fulfills the listed property and (-) and (- -) indicate that it does not.

Prospect for application in power systems and control

The present thesis has mainly applications from power systems and optimal control in mind. Characteristic in these two classes of problems are *non-convex constraints* which essentially leave two classes of distributed algorithms for further investigation: primal-dual methods and internal decomposition methods. As ALADIN is a mix between these two it seems to be an excellent fit. However, as indicated above, ALADIN has a quite heavy coordination step and requires a quite large amount of central communication. Motivated by this fact, a main focus of the present thesis will be to develop a *decentralized* ALADIN-based algorithm preserving its favorable convergence guarantees and *reducing its communication requirements*. In the next chapter, we present a new algorithmic framework fulfilling these requirements which we call *bi-level ALADIN*.

[6] With advanced features we mean features such as asynchronous operation, the ability to handle communication delays, operation over directed graphs or a changing graph topology.

4 Bi-level Distributed ALADIN

In the previous sections we highlighted that the core advantages of ALADIN are its fast local convergence guarantees, and an efficient handling of non-convex constraints. Disadvantages are a large amount of communication and a missing decentralization. This chapter proposes a framework in which both is possible—operation based on neighbor-to-neighbor communication (decentralized optimization) and reducing the communication overhead while preserving ALADIN's fast local convergence guarantees.

By having a closer look at Algorithm 5, one can observe that only the coordination step 2) requires central coordination and global communication.[1] Thus, the idea of this chapter is decentralize this step by means of decentralized inner algorithms.

Large parts of this section are based on [Eng+20; Eng+19b]. The sparsity encoding techniques and the decentralized conjugate gradient algorithm are improved versions from [Eng+20] and not published so far. The convergence analysis in Section 4.5 is a unified version of [Eng+20], which does not require any results from [HFD16] and is purely based on nonlinear programming theory from Chapter 2. Similar approaches can also be found in the literature on numerical methods for optimal control [Kou+16; FSD15].

4.1 Reducing communication by condensing

In a first step, we apply so-called reduced-space or condensing techniques to the coordination QP of ALADIN. By doing so, we reduce the dimension of the

[1] A globalization also requires either central coordination or costly additional communication. However, as stated before, here we focus on a *local* algorithm. Designing distributed globalization routines is still an open problem.

QP and thus ALADIN's communication overhead. The basic ideas for doing so are from the literature for nonlinear programming [NW06].

Recall the coordination QP of ALADIN including slack variables (2.25)

$$\min_{\Delta x, d} \sum_{i \in \mathcal{R}} \frac{1}{2} \Delta x_i^\top B_i^k \Delta x_i + \nabla f_i^\top(x_i^k) \Delta x_i + \lambda^{k\top} d + \frac{\rho}{2} \|d\|_2^2 \tag{4.1a}$$

$$\text{subject to} \qquad C_i^k \Delta x_i = 0 \qquad \forall i \in \mathcal{R}, \tag{4.1b}$$

$$\sum_{i \in \mathcal{R}} A_i (x_i^k + \Delta x_i) - b = d \qquad | \ \lambda^{\text{QP}}. \tag{4.1c}$$

The first technique we apply is the so-called nullspace-method [NW06, Ch. 12] eliminating the constraint linearizations $C_i^k \in \mathbb{R}^{n_{hi}^k \times n_{xi}}$ from (4.1b). To this end, we need a basis of the nullspace of C_i^k. Let us construct a matrix $Z_i^k \in \mathbb{R}^{n_{xi} \times (n_{xi} - n_{hi}^k)}$ with $C_i^k Z_i^k = 0$ for all $i \in \mathcal{R}$ forming a basis of this nullspace with its columns.[2] This allows parameterizing $\Delta x_i = Z_i^k \Delta v_i$ and thus

$$\min_{\Delta v, d} \sum_{i \in \mathcal{R}} \frac{1}{2} \Delta v_i^\top \bar{B}_i^k \Delta v_i + \bar{g}_i^{k\top} \Delta v_i + \lambda^{k\top} d + \frac{\nu}{2} \|d\|_2^2 \tag{4.2a}$$

$$\text{subject to} \sum_{i \in \mathcal{R}} A_i x_i^k + \bar{A}_i^k \Delta v_i - b = d \qquad | \ \lambda^{\text{QP}}. \tag{4.2b}$$

In case of many active constraints, also the reduced Hessian $\bar{B}_i^k := Z_i^{k\top} B_i^k Z_i^k$, the reduced gradient $\bar{g}_i^k := g_i^{k\top} Z_i^k$ and the reduce coupling matrices $\bar{A}_i^k = A_i Z_i^k$ are much smaller in dimension compared with (4.1). We also reduce dimension of the decision vector Δx_i from n_{xi} to $n_{xi} - n_{hi}^k$. Hence, we reduce communication when solving (4.2) instead of (4.1) in step 2) of ALADIN. A detailed communication analysis will be given in Section 4.6.

[2] If rows of C_i^k are linearly dependent though, the number of rows of Z_i^k is $n_{xi} - n_{hi}^k + d_i^k$, where d_i^k is the number of linearly dependent rows of C_i^k.

The Schur complement

Employing the Schur-complement allows to further reduce the dimensionality of the QP (4.2). For doing so, we need form the inverse of $\bar{B}_i^k$. Hence, we assume the following.

Assumption 1 The reduced Hessian approximation $\bar{B}_i^k$ is symmetric and positive definite for all iterates $k \in \mathbb{N}$ and all $i \in \mathcal{R}$.

This assumption holds for example if $\nabla_{xx}\mathcal{L}(x^k, \lambda^k, \rho^k)$ is positive definite on the range space spanned by $\nabla h(x^k)$ for all $k \in \mathbb{N}$. However, it also holds by choosing an appropriate Hessian approximation $\bar{B}_i^k$ such that the assumption holds. We will further comment on how to do so in Chapter 6. Moreover, we assume the following.

Assumption 2 The matrix $A = [A_1, \ldots, A_R]$ has full row rank.

As $B_i^k \succ 0$, the KKT conditions (2.2) for problem (4.2) are necessary and sufficient for a global minimizer to (4.2). The Lagrangian to (4.2) is

$$\mathcal{L}^{\mathrm{QP}}(\Delta v, d, \lambda^{\mathrm{QP}}) := \sum_{i\in\mathcal{R}} \frac{1}{2}\Delta v_i^\top \bar{B}_i^k \Delta v_i + \bar{g}_i^{k\top}\Delta v_i + \lambda^{k\top} d + \frac{\rho}{2}\|d\|_2^2 + \lambda^{\mathrm{QP}\top}\left(\sum_{i\in\mathcal{R}} A_i x_i^k + \bar{A}_i^k \Delta v_i - b - d\right).$$

Thus, the KTT conditions $\nabla\mathcal{L}^{\mathrm{QP}}(\Delta v, d, \lambda^{\mathrm{QP}}) = 0$ read

$$\bar{B}^k \Delta v + \bar{g}^k + \bar{A}^{k\top}\lambda^{\mathrm{QP}} = 0, \tag{4.3a}$$

$$\lambda^k + \rho d - \lambda^{\mathrm{QP}} = 0, \tag{4.3b}$$

$$A^k x^k + \bar{A}^k \Delta v - b - d = 0, \tag{4.3c}$$

where $\bar{B}^k = \mathrm{diag}_{i\in\mathcal{R}}(\bar{B}_i^k)$, $\bar{g}^k = (\bar{g}_1^{k\top}, \ldots, \bar{g}_1^{R\top})^\top$, $\bar{A} = (\bar{A}_1, \ldots, \bar{A}_R)$ and $\Delta v = (\Delta v_1^\top, \ldots, \Delta v_R^\top)^\top$. Here, $\mathrm{diag}_{i\in\mathcal{R}}(H_i)$ denotes diagonal concatenation of

matrices H_i indexed by the set $\mathcal{R}$. By eliminating (4.3b) with $s = \frac{1}{\rho}(\lambda^{\mathrm{QP}} - \lambda^k)$, we obtain

$$\bar{B}^k \Delta v + \bar{g}^k + \bar{A}^{k\top} \lambda^{\mathrm{QP}} = 0, \tag{4.4a}$$

$$A^k x^k + \bar{A}^k \Delta v - b - \frac{1}{\rho}(\lambda^{\mathrm{QP}} - \lambda^k) = 0. \tag{4.4b}$$

In order to further reduce dimensionality of (4.4), we rearrange (4.4a) as

$$\Delta v = -\bar{B}^{k^{-1}} (\bar{g}^k + \bar{A}^{k\top} \lambda^{\mathrm{QP}}), \tag{4.5}$$

where we used Assumption 1. Inserting to (4.4b) yields

$$\left(\underbrace{\bar{A}^k \, \bar{B}^{k^{-1}} \bar{A}^{k\top}}_{:=S^k} + \frac{1}{\rho} I \right) \lambda^{\mathrm{QP}} = \underbrace{A^k x^k - \bar{A}^k \bar{B}^{k^{-1}} \bar{g}^k}_{:=s^k} - b + \frac{1}{\rho} \lambda^k. \tag{4.6}$$

Exploiting block structure

Because of the block structure of $\bar{B}^k$ and $\bar{A}$ in (4.6), one can write

$$S^k = \bar{A}^k \, \bar{B}^{k^{-1}} \bar{A}^{k\top} = \sum_{i \in \mathcal{R}} \bar{A}_i^k \, \bar{B}_i^{k^{-1}} \bar{A}_i^{k\top} = \sum_{i \in \mathcal{R}} S_i^k \tag{4.7}$$

with $S_i^k := \bar{A}_i^k \, \bar{B}_i^{k^{-1}} \bar{A}_i^{k\top}$. Moreover,

$$s^k = A x^k - \bar{A}^k \bar{B}^{k^{-1}} \bar{g}^k = \sum_{i \in \mathcal{R}} A_i x_i^k - \bar{A}_i \bar{B}_i^{k^{-1}} \bar{g}_i^k = \sum_{i \in \mathcal{R}} s_i^k \tag{4.8}$$

with $s_i^k := A_i^k x_i^k - \bar{A}_i \bar{B}_i^{k^{-1}} \bar{g}_i^k$. Observe that the variables S_i^k and s_i^k can be computed entirely locally the subsystems $i \in \mathcal{R}$. Hence, it suffices that the subsystems communicate these two quantities to a coordinator, which then solves (4.6).

Note that (4.6) is a linear system of equations of dimension of the number of consensus constraints n_c, which is typically much smaller than the total number of variables $n_x + n_g + n_c$. If one solves (4.6) for λ^{QP}, all Δv_i and Δx_i can be computed by back substitution in (4.5) and by $\Delta x_i^k = Z_i^k \Delta v_i^k$.

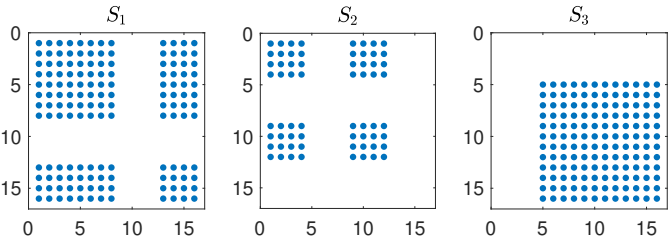


Figure 4.1: Sparsity patterns of $\{S_i\}_{i\in\{1,2,3\}}$ for the 5-bus system (Figure 5.2b).

In summary, this subsection showed that instead of solving (4.1) one can solve (4.6) in the coordination step 2) of ALADIN (Algorithm 5). This reduces communication and coordination drastically in case of problems with many active constraints. In this case, it suffices to communicate sensitivities of reduced dimension (Schur complements) S_i^k and vectors s_i^k instead of the full sensitivities. However, note that although this version of ALADIN requires less communication, ALADIN still requires central coordination. In the next subsection we derive an algorithm solving (4.6) in a *decentralized* fashion rendering ALADIN a decentralized optimization algorithm.

4.2 Decentralization of coordination

Next we develop two algorithms solving the reduced coordination step (4.6) in a *decentralized* fashion, which means purely based on *neighbor-to-neighbor* communication. For doing so, we first analyze the sparsity pattern of S_i.

The sparsity of S_i

Although the matrix S is generally dense, the matrices S_i are not necessarily. Figure 4.1 shows the sparsity patterns from an optimization problem from power systems which we will use in Chapter 5. Here, certain rows/columns seem to be structurally zero. This is the case since often the rows of A_i express coupling between variables in *two* different subsystems. The zero

rows/columns in S_i come from the *zero rows* in A_i of all *remaining subsystems*. In the following we call the two subsystems $i, j \in \mathcal{R}$, which share a common non-zero row in their A_i *assigned* to this row.

Definition 6 A subsystem $i \in \mathcal{R}$ is called assigned to consensus constraint $c \in C = \{1, \dots, n_c\}$ if the cth row of A_i is non-zero.

Moreover, we collect all subsystems assigned to consensus constraint $c \in C$ in

$$\mathcal{R}(c) := \{i \in \mathcal{R} \mid i \text{ assigned to } c \in C\},$$

and we collect all consensus constraints assigned to a subsystem $i \in \mathcal{R}$ in

$$C(i) := \{c \in C \mid i \in \mathcal{R}(c)\}.$$

With that, we are ready to make a statement on the sparsity of S_i.

Lemma 1 The rows and columns of S_i and the entries of s_i with $c \notin C(i)$ are zero.

Proof. By Definition 6, all rows of $\bar{A}_i^k = A_i Z_i^k$ with $c \notin C(i)$ are zero. It follows immediately that the rows and columns of $S_i^k = \bar{A}_i^k \, \bar{B}_i^{k^{-1}} \bar{A}_i^{k\top}$ and the entries of $s_i^k = A_i^k x_i^k - \bar{A}_i \bar{B}_i^{k^{-1}} \bar{g}_i^k$ with $c \notin C(i)$ are zero. □

4.2.1 Exploiting sparsity for decentralization

The idea now is to exploit this sparsity for decentralization. Our goal is to formulate (4.6) as

$$\left(\sum_{i \in \mathcal{R}} \tilde{S}_i^k \right) \lambda^{\mathrm{QP}} = \sum_{i \in \mathcal{R}} \tilde{s}_i^k, \tag{4.9}$$

where $\tilde{S}_i^k$ and $\tilde{s}_i^k$ are formed entirely locally and preserve the sparsity of S_i^k and s_i^k. Two quantities hindering us are $\frac{1}{\rho} I$ and $(\frac{1}{\rho}\lambda^k - b)$ in (4.6) which do not "belong" to any of the subsystems.

The main idea in the following is, to "distribute" $\frac{1}{\rho}I$ and $(\frac{1}{\rho}\lambda^k - b)$ to subproblems, which are assigned to the corresponding consensus constraint. To this end, we define

$$\tilde{S}_i^k := S_i^k + \sum_{c=1}^{n_c} \frac{\iota_{\mathcal{C}(i)}(c)}{|\mathcal{R}(c)|\,\rho} I_c, \quad \text{and} \quad \tilde{s}_i^k := s_i^k + \sum_{c=1}^{n_c} \frac{\iota_{\mathcal{C}(i)}(c)}{|\mathcal{R}(c)|\,\rho} (\lambda_c^k - b_c) e_c. \tag{4.10}$$

where I_c is a zero matrix except for $[I]_{cc} = 1$, $\iota_C(\cdot)$ is the indicator function of the set C, and $e_c \in \mathbb{R}^{n_c}$ is the cth unit vector. This way, the sparsity patterns of S_i^k and s_i^k are preserved and (4.6) is reformulated in form of (4.9).

This sparsity pattern can in turn be exploited by ADMM leading to a *decentralized ADMM*. To this end, we express (4.9) as a strongly convex QP.

Lemma 2 The QP

$$\min_{\bar{\lambda}} \ \frac{1}{2}\bar{\lambda}^\top \sum_{i=1}^{R} \tilde{S}_i \bar{\lambda} - \sum_{i=1}^{R} \tilde{s}_i^\top \bar{\lambda} \tag{4.11}$$

is strongly convex, it's minimizer is unique and solves (4.9).

Proof. First, we prove strong convexity of (4.11), i.e. $\tilde{S} := \sum_{i=1}^{R} \tilde{S}_i \succ 0$. By Assumption 1 we have $\bar{B}_i^k \succ 0$ and hence $\bar{B}^k \succ 0$. Thus, $\bar{B}^{k^{-1}} \succ 0$. Furthermore, by Z^k having full column rank, $\bar{A}^k = AZ^k$ has full row rank by Assumption 2. Hence, $y := \bar{A}^k x \neq 0$ for all $x \neq 0$ and thus $S^k = \bar{A}^k\, \bar{B}^{k^{-1}} \bar{A}^{k\top} \succ 0$ by $y^\top \bar{B}^{k^{-1}} y > 0$. By the definition of $\tilde{S}$, (4.10), we have $\tilde{S} = S + \frac{1}{\rho} I \succ 0$ by $S \succ 0$.

Let us show that $\tilde{S}_i = \tilde{S}_i^\top$. We have $S_i^\top = (\bar{A}^k\, \bar{B}^{k^{-1}} \bar{A}^{k\top})^\top = \bar{A}^k\, (\bar{B}^{k^{-1}})^\top \bar{A}^{k\top} = \bar{A}^k\, (\bar{B}^{k^\top})^{-1} \bar{A}^{k\top}$. By Assumption 1, we have $\bar{B}^{k^\top} = \bar{B}^k$ and hence the assertion follows.

It remains to show that the minimizer of (4.11) is unique and coincides with the solution to (4.9). Uniqueness follows from strong convexity. Again from strong convexity, the first-order necessary conditions are also sufficient for a minimum. They read $\frac{1}{2}(\sum_{i=1}^{R} \tilde{S}_i + \sum_{i=1}^{R} \tilde{S}_i^\top)\bar{\lambda} - \sum_{i=1}^{R} \tilde{s}_i = 0$ and thus coincide with (4.9) by symmetry of $\tilde{S}$. □

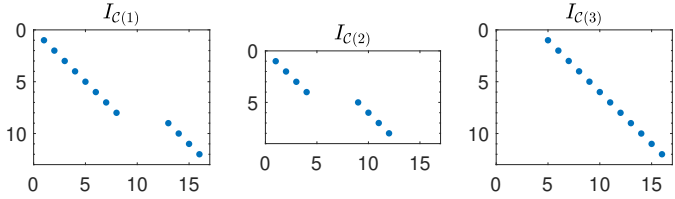


Figure 4.2: Sparsity patterns for $\{I_{C(i)}\}_{i\in\{1,2,3\}}$.

4.2.2 Decentralized ADMM

To derive a decentralized variant of ADMM (d-ADMM), we restate (4.11) in so-called *consensus form* (A.9). We introduce matrices $I_{C(i)} \in \mathbb{R}^{|C(i)|\times n_c}$ mapping from global Lagrange multipliers $\bar{\lambda} \in \mathbb{R}^{n_c}$ to local ones $\lambda_i \in \mathbb{R}^{|C(i)|}$ by $\lambda_i := I_{C(i)}\bar{\lambda}$. These matrices are defined as the horizontal concatenation of unit vectors $e_c \in \mathbb{R}^{n_c}$ indexed by the set $\mathcal{R}$ as $I_{\mathcal{R}} := \left(e_s^\top\right)_{s\in\mathcal{R}} \in \mathbb{R}^{|\mathcal{R}|\times n_c}$. Alternatively, these matrices may be viewed as identity matrices, where certain rows have been "deleted" eliminating zero rows/columns in $\tilde{S}_i$ and $\tilde{s}_i$. Figure 4.2 shows exemplary sparsity patterns of $\{I_{C(i)}\}_{i\in\mathcal{R}}$ for an example from power systems from Section B.4. The matrices $I_{C(i)}$ have an interesting property, namely that $\tilde{S}_i$ and $\tilde{s}_i$ are invariant under multiplication with $I_{C(i)}^\top I_{C(i)}$.

Lemma 3 (Invariance of $\tilde{S}_i$ and $\tilde{s}_i$ under multiplication with $I_{C(i)}^\top I_{C(i)}$) It holds that $I_{C(i)}^\top I_{C(i)}\tilde{S}_i = \tilde{S}_i$ and $I_{C(i)}^\top I_{C(i)}\tilde{s}_i = \tilde{s}_i$.

Proof. By definition, $I_{C(i)}$ is composed of unit vectors for which $e_i^\top e_j = 1$, if and only if $i = j$ and $e_i^\top e_j = 0$ else. Thus, $I_{C(i)}^\top I_{C(i)}$ is an identity matrix with zero rows/columns for all rows/columns indexed by $c \notin C(i)$. As these rows are anyways zero in $\tilde{S}_i$ and $\tilde{s}_i$ by Lemma 1, the assertion follows. □

Algorithm 6. d-ADMM for problem (4.12)

Initialization: $\bar{\lambda}_i^0, \gamma_i^0$ **for all** $i \in \mathcal{R}$, ρ
Repeat for all $i \in \mathcal{R}$**:**

1) $\lambda_i^{n+1} = \left(\hat{S}_i^k + \rho I\right)^{-1} \left(\hat{s}_i^k - \gamma_i^n + \rho \bar{\lambda}_i^n\right)$ (local)

2) $\bar{\lambda}_i^{n+1} = (\Lambda_i)^{-1} \sum_{j \in N(i)} I_{ij} \lambda_j^{n+1}$ (neighbor-to-neighbor)

3) $\gamma_i^{n+1} = \gamma_i^n + \rho(\lambda_i^{n+1} - I_{C(i)} \bar{\lambda}^{n+1})$ (local)

With that, we are able to reformulate (4.11) in consensus form. Consider $f_i(\bar{\lambda}) := \frac{1}{2} \bar{\lambda}^\top \tilde{S}_i \bar{\lambda} - \tilde{s}_i^\top \bar{\lambda}$ as local objectives. By Lemma 3, $f_i(\bar{\lambda}) = f_i(I_{C(i)}^\top I_{C(i)} \bar{\lambda}) = f_i(I_{C(i)}^\top \lambda_i)$. Thus, problem (4.11) can be written as

$$\begin{aligned} \min_{\lambda_1, \ldots, \lambda_R, \bar{\lambda}} \quad & \sum_{i \in \mathcal{R}} f_i(I_{C(i)}^\top \lambda_i) \\ \text{subject to} \quad & \lambda_i = I_{C(i)} \bar{\lambda} \;\mid\; \gamma_i \text{ for all } i \in \mathcal{R}, \end{aligned} \tag{4.12}$$

with Lagrange multipliers $\gamma_i \in \mathbb{R}^{|C(i)|}$.

Solution via ADMM

Let us evaluate the ADMM iterations (Algorithm 3) for problem (4.12). The Augmented Lagrangian for (4.12) is

$$\begin{aligned} &\mathcal{L}^\rho(\lambda_1, \ldots, \lambda_R, \bar{\lambda}, \gamma_1, \ldots, \gamma_R) = \\ &\qquad \sum_{i \in \mathcal{R}} f_i(I_{C(i)}^\top \lambda_i) + \gamma_i^\top (\lambda_i - I_{C(i)} \bar{\lambda}) + \frac{\rho}{2} \|(\lambda_i - I_{C(i)} \bar{\lambda})\|_2^2. \end{aligned} \tag{4.13}$$

Necessary conditions for minimizing (4.13) with respect to $\lambda_1, \ldots, \lambda_R$ are

$$\hat{S}_i^k \lambda_i^{n+1} - \hat{s}_i^k + \gamma_i^n + \rho(\lambda_i^{n+1} - \bar{\lambda}_i^n) = 0, \quad \text{for all } i \in \mathcal{R},$$

where $\hat{S}_i^k := I_{C(i)} \tilde{S}_i^k I_{C(i)}^\top$ and $\hat{s}_i^k := I_{C(i)} s_i^k$. Rearranging terms yields step 1) of Algorithm 6.

Step 2) of Algorithm 3 minimizes (4.13) with respect to $\bar{\lambda}$. The first-order optimality conditions are

$$\sum_{i\in\mathcal{R}} -I_{C(i)}^{\top}\gamma_i^n - \rho I_{C(i)}^{\top}(\lambda_i^{n+1} - I_{C(i)}\bar{\lambda}^{n+1}) = 0. \tag{4.14}$$

Step 3) of Algorithm 3 is

$$\gamma_i^{n+1} = \gamma_i^n + \rho(\lambda_i^{n+1} - I_{C(i)}\bar{\lambda}^{n+1}). \tag{4.15}$$

Summing up for all $i \in \mathcal{R}$ yields

$$\sum_{i\in\mathcal{R}} \gamma_i^{n+1} = \sum_{i\in\mathcal{R}} \gamma_i^n + \rho(\lambda_i^{n+1} - I_{C(i)}\bar{\lambda}^{n+1}).$$

Multiplying by $I_{C(i)}^{\top}$ from the left together with (4.14) implies that $\sum_{i\in\mathcal{R}} I_{C(i)}^{\top}\gamma_i^{n+1} = 0$ after the first ADMM iteration. Hence (4.14) becomes

$$\sum_{i\in\mathcal{R}} I_{C(i)}^{\top}\lambda_i^{n+1} - I_{C(i)}^{\top}I_{C(i)}\bar{\lambda}^{n+1} = 0. \tag{4.16}$$

Multiplying by $I_{C(j)}$ from the left yields

$$\sum_{i\in\mathcal{R}} \underbrace{I_{C(j)}I_{C(i)}^{\top}}_{:=I_{ji}} \lambda_i^{n+1} - I_{C(j)} \underbrace{\sum_{i\in\mathcal{R}} I_{C(i)}^{\top}I_{C(i)}}_{:=\Lambda} \bar{\lambda}^{n+1} = 0. \tag{4.17}$$

Since $I_{C(j)}I_{C(j)}^{\top} = I$ and $AB = BA$ for diagonal matrices A, B, we have

$$I_{C(j)}\Lambda\bar{\lambda}^{n+1} = I_{C(j)}I_{C(j)}^{\top}I_{C(j)}\Lambda\bar{\lambda}^{n+1} = I_{C(j)}\Lambda I_{C(j)}^{\top}I_{C(j)}\bar{\lambda}^{n+1} = \Lambda_j\bar{\lambda}_j^{n+1},$$

where we define $\Lambda_i := I_{C(i)}\Lambda I_{C(i)}^{\top}$. Hence, (4.17) can be written as

$$\bar{\lambda}_j^{n+1} = \left(\Lambda_j\right)^{-1} \sum_{i\in\mathcal{R}} I_{ji}\lambda_i^{n+1}.$$

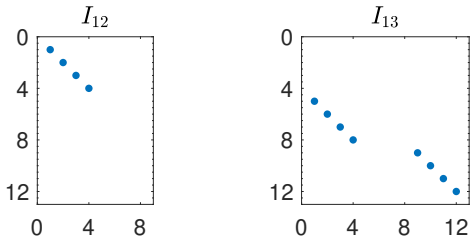


Figure 4.3: Coupling matrices I_{12} and I_{13} corresponding to the matrices $\{I_{C(i)}\}_{i\in\{1,2,3\}}$ shown in Figure 4.2.

As $I_{ji} = 0$ for $j \notin N(i)$, we can replace $i \in \mathcal{R}$ by $i \in N(j)$, where we define the *neighborhood* $N(i)$ of subsystem $i \in \mathcal{R}$ by

$$N(i) := \{j \in \mathcal{R} \mid \text{there exists a } c \in C \text{ with } i, j \in \mathcal{R}(c)\}.$$

The identity $I_{ji} = 0$ for $j \notin N(i)$ follows from this definition and the definition of $I_{C(i)}$. This yields step 2) of Algorithm 6.

Note that in Algorithm 6 we have entirely local steps (step 1) and step 3)) and one neighbor-to-neighbor step (step 2)). This means that the algorithm is *decentralized*, i.e. all communication is done locally between neighbors and a central entity is *not* required. Superscripts $(\cdot)^n$ denote d-ADMM iterates. Quantities with superscripts $(\cdot)^k$ do not change during the d-ADMM iterations—they refer to (outer) ALADIN iterations. Furthermore, note that the initial guesses for $\bar{\lambda}_i^0$ have to be *consistent*, i.e. they have to satisfy $\bar{\lambda}_i^0 = I_{C(i)}\bar{\lambda}^0$ for all $i \in \mathcal{R}$.

Observe that the matrices $I_{ij} \in \mathbb{R}^{|C(i)|\times|C(j)|}$ encode "coupling information" between the subsystems. More specifically, they define which entries of the vectors λ_i, and in subsystem $i \in \mathcal{R}$ belong which entry of the vectors λ_j in subsystem $j \in \mathcal{R}$. Furthermore, observe that $I_{ii} = I$. The sparsity patterns of I_{12} and I_{13} corresponding to $I_{C(1)}$, $I_{C(2)}$, $I_{C(3)}$ from Figure 4.2 are illustrated in Figure 4.3. One can observe that subsystem 1 only requires the first four entries of λ_2 from subsystem 2.

d-ADMM is guaranteed to converge to the minimizer of (4.12) and the minimizer of (4.12) corresponds to the solution of (4.9) by Lemma 2. Hence, we

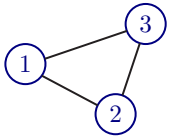


Figure 4.4: Communication graph for our power systems example.

achieved our goal of deriving a decentralized version in the coordination step of ALADIN.

Remark 10 (Global variable consensus vs. general consensus) An alternative way of reformulating (4.11) would be via the (simpler) so-called *global variable* consensus [Boy+11]

$$\begin{aligned} \min_{\lambda_1,\ldots,\lambda_R,\bar{\lambda}} \quad & \sum_{i\in\mathcal{R}} f_i(\lambda_i) \\ \text{subject to} \quad & \lambda_i = \bar{\lambda} \;\mid\; \gamma_i, \text{ for all } i \in \mathcal{R}. \end{aligned} \tag{4.18}$$

as we did in [Eng+20]. However, this leads to the situation that $f = \sum_{i\in\mathcal{R}} f_i$ loses it's strict convexity property with respect to the inflated variable block $\lambda^\top = (\lambda_1^\top, \ldots, \lambda_R^\top)$ if not all f_i depend on all components of λ_i as we have here because of the sparsity of $\tilde{S}_i$. This is *not* the case in (4.12) since we eliminate locally redundant variables by means of the matrices $I_{C(i)}$. This is important since convexity vs. strict convexity can lead to sublinear vs. linear convergence for ADMM [MO17; HY12; GB16; HL17; Shi+14].

Remark 11 (Communication graph) One can regard the set of subsystems $\mathcal{R}$ as nodes and the set $\mathcal{E} := \{(i,j) \in \mathcal{R}\times\mathcal{R} \mid \text{there exists a } c \in \mathcal{C} \text{ with } i,j \in \mathcal{R}(c)\}$ as edges in a communication graph $G = (\mathcal{N}, \mathcal{E})$. The edges of this graph can equivalently be characterized by all (i,j) for which $I_{ij} \neq 0$. For the example from power systems from Section B.4, the graph is fully connected and displayed in Figure 4.4.

4.2.3 Decentralized conjugate gradient

Using d-ADMM as an inner coordination algorithm has the advantage that only very few information has to be exchanged between neighboring subsystems, namely the average primal values $\bar{\lambda}$. However, ADMM requires tuning which might be difficult (cf. Section 5.5) and the convergence rate is at most linear.

To address these drawbacks, the idea of this section is to develop a decentralized variant of the *conjugate gradient* algorithm (d-CG) with fast convergence. A main advantage of CG is, its convergence guarantee in *at most* n_c *steps* to the *exact* solution of a linear system of equations [NW06, Ch. 5.1]. Although this property is weakened in practice due to numerical round-off errors, the practically observed convergence rate is superlinear, which is very fast compared with other methods such as ADMM [BK01].

The idea of CG is to iteratively minimize (4.11) by generating iterates $\{\bar{\lambda}^n\}$ which are $\tilde{S}$*-conjugate* to each other, i.e. $\bar{\lambda}^{n\top}\tilde{S}\bar{\lambda}^m = 0$ for all $n \neq m$. $\tilde{S}$-conjugacy can be seen as a generalization of orthogonality. Performing a one-dimensional minimizations along these conjugate directions $\{\bar{\lambda}^n\}$ yields the conjugate gradient algorithm [NW06, Ch. 5.1]. The iterations are given by

$$\alpha^n = \frac{r^{n\top} r^n}{p^{n\top}\tilde{S}p^n}, \tag{4.19a}$$

$$\bar{\lambda}^{n+1} = \bar{\lambda}^n + \alpha^n p^n, \tag{4.19b}$$

$$r^{n+1} = r^n - \alpha^n \tilde{S}p^n, \tag{4.19c}$$

$$\beta^n = \frac{r^{n+1\top} r^{n+1}}{r^{n\top} r^n}, \tag{4.19d}$$

$$p^{n+1} = r^{n+1} + \beta^n p^k, \tag{4.19e}$$

where the initialization has satisfy $r^0 = p^0 = \tilde{s} - \tilde{S}\bar{\lambda}^0$. Now, let us again use the sparsity results from Lemma 1, Lemma 3 and the following additional result.

Lemma 4 It holds that $\sum_{i\in\mathcal{R}} I_{C(i)}^\top \Lambda_i^{-1} I_{C(i)} = I$.

Proof. Since Λ is diagonal, and by definition of $I_{C(i)}$ as the concatenation of unit vectors, we have $\Lambda_i^{-1} = (I_{C(i)}\Lambda I_{C(i)}^\top)^{-1} = I_{C(i)}\Lambda^{-1} I_{C(i)}^\top$.

Combination with Lemma 3 and $AB = BA$ for diagonal matrices yields $\sum_{i\in\mathcal{R}} I_{C(i)}^\top \Lambda_i^{-1} I_{C(i)} = \sum_{i\in\mathcal{R}} I_{C(i)}^\top I_{C(i)} \Lambda^{-1} I_{C(i)}^\top I_{C(i)} =$
$\sum_{i\in\mathcal{R}} I_{C(i)}^\top I_{C(i)} I_{C(i)}^\top I_{C(i)} \Lambda^{-1} = \sum_{i\in\mathcal{R}} I_{C(i)}^\top I_{C(i)} \Lambda^{-1} = \Lambda\,\Lambda^{-1} = I.$ □

The main idea in decentralized conjugate gradient is to exploit sparsity in (4.19a)-(4.19e) by the mappings $I_{C(i)}$ which we used already for d-ADMM. Let us start by defining—similar to d-ADMM—local equivalents for all involved variables $\bar{\lambda}, r$ and p

$$\lambda_i := I_{C(i)}\bar{\lambda}, \qquad r_i := I_{C(i)} r, \quad \text{and} \quad p_i := I_{C(i)} p. \tag{4.20}$$

With these definitions, we decompose (4.19a). For doing so, we define $\alpha^n = \frac{\eta^n}{\sigma^n}$ and $\eta^n := r^{n\top} r^n$. The idea is to write this identity in terms of r_i^n instead of r^n. By Lemma 4, we write η as

$$r^{n\top} r^n = r^{n\top} I\, r^n = r^{n\top} \Big(\sum_{i\in\mathcal{R}} I_{C(i)}^\top \Lambda_i^{-1} I_{C(i)} \Big)\, r^n.$$

Defining $\eta_i := r_i^{n\top} \Lambda_i^{-1} r_i^n$ and by using (4.20) we obtain

$$\eta = r^{n\top} r^n = \sum_{i\in\mathcal{R}} r_i^{n\top} \Lambda_i^{-1} r_i^n = \sum_{i\in\mathcal{R}} \eta_i,$$

where η_i can be computed locally. However, note that the sum $\sum_{i\in\mathcal{R}} \eta_i$ requires *global communication* of one scalar η_i per subsystem. This yields step 3) (a) and step 4) of Algorithm 7. Next, we decompose the denominator $\sigma^n := p^{n\top} \tilde{S} p^n$ in (4.19a). By the definition of $\tilde{S}$, p_i and by Lemma 3, the denominator can be written as

$$\sigma^n = p^{n\top} \sum_{i\in\mathcal{R}} \tilde{S}_i p^n = p^{n\top} \sum_{i\in\mathcal{R}} I_{C(i)}^\top I_{C(i)} \tilde{S}_i I_{C(i)}^\top I_{C(i)} p^n = \sum_{i\in\mathcal{R}} p_i^{n\top} \hat{S}_i p_i^n = \sum_{i\in\mathcal{R}} \sigma_i^n$$

This yields step 1) and step 5) (c) of Algorithm 7. Let us consider (4.19b) and (4.19e). They are entirely local steps since multiplying both equations by $I_{C(i)}$ from the left yields

$$\bar{\lambda}_i^{n+1} = \bar{\lambda}_i^n + \frac{\eta^n}{\sigma^n} p_i^n \quad \text{and} \quad p_i^{n+1} = r_i^{n+1} + \frac{\eta^{n+1}}{\eta^n} p_i^k$$

Algorithm 7. d-CG for problem (4.12)

Initialization: λ^0 **and** $r^0 = p^0 = \tilde{s} - \tilde{S}\lambda^0$

Repeat for all $i \in \mathcal{R}$**:**

1) $\sigma^n = \sum_{i\in\mathcal{R}} \sigma_i^n$ (scalar global sum)

2) $r_i^{n+1} = r_i^n - \frac{\eta^n}{\sigma^n} \sum_{j\in N(i)} I_{ij} u_j^n$ (neighbor-to-neighbor)

3) (a) $\eta_i^{n+1} = r_i^{n+1} \Lambda_i^{-1} r_i^{n+1}$ (b) $\lambda_i^{n+1} = \lambda_i^n + \frac{\eta^n}{\sigma^n} p_i$ (local)

4) $\eta^{n+1} = \sum_{i\in\mathcal{R}} \eta_i^{n+1}$ (scalar global sum)

5) (a) $p_i^{n+1} = r_i^{n+1} + \frac{\eta^{n+1}}{\eta^n} p_i^n$ (b) $u_i^{n+1} = \hat{S}_i p_i^{n+1}$ (c) $\sigma_i^{n+1} = p_i^{n+1\top} \hat{S}_i p_i^{n+1}$ (local)

comprising step 5) (a) and step 3) (b) of Algorithm (7). In a last step, we need to decompose (4.19c) which requires neighbor-to-neighbor communication due to the product $\tilde{S}p^n$ as we shall see. Again, by Lemma 3 and by the definitions of p_i and $\hat{S}_j$, we have

$$r^{n+1} = r^n - \frac{\eta^n}{\sigma^n} \tilde{S} p^n = r^n - \frac{\eta^n}{\sigma^n} \sum_{j\in\mathcal{R}} \tilde{S}_j p^n = r^n - \frac{\eta^n}{\sigma^n} \sum_{j\in\mathcal{R}} I_{C(j)}^\top \hat{S}_j p_j^n.$$

Multiplying by $I_{C(i)}$ from the left yields

$$r_i^{n+1} = r_i^n - \frac{\eta^n}{\sigma^n} \sum_{j\in N(i)} I_{ij} \hat{S}_j p_j^n = r_i^n - \frac{\eta^n}{\sigma^n} \sum_{j\in N(i)} I_{ij} u_j^n,$$

where we recall that $I_{ij} := I_{C(i)} I_{C(j)}^\top$ and $u_j^n := \hat{S}_j p_j^n$. Note that we only sum over the neighbors of subsystem i as $I_{ij} = 0$ if $j \neq N(i)$. Moreover, the quantities $u_j^n = \hat{S}_j p_j^n$ can again be computed locally by each subsystem. This yields step 2) and step 5) (b) of Algorithm 7.

Remark 12 (Decentralized operation via hopping) Note that although for computing η and σ global communication is required, this communication may also be executed in decentralized fashion via "hopping". By that we mean that each subsystem adds it's η_i to the sum which was received from a neighbor until each subsystem was reached. By a second "hopping" round, the sum can then be "broadcasted" to all subsystems. In general, such an approach is

somewhat problematic as it requires all subsystems to wait until one hopping round has to be performed, which can potentially take a long time in case of bad local communication links as these delays sum up. However, as in d-CG we only have *one scalar* per subsystem independently of the problem size, this approach might be justified.

4.2.4 Consistent initialization

Note that d-CG requires a consistent initialization satisfying $r^0 = p^0 = \tilde{s} - \tilde{S}\lambda^0$. Multiplying this equation by $I_{C(i)}$ from the left and using $I_{C(i)}^\top I_{C(i)} \tilde{S}_i = \tilde{S}_i$ yields

$$r_i^0 = p_i^0 = \sum_{j \in \mathcal{R}} I_{C(i)} \tilde{s}_j - \sum_{j \in \mathcal{R}} I_{C(i)} \tilde{S}_j \lambda^0 = \sum_{j \in \mathcal{R}} I_{ij} \hat{s}_j - \sum_{j \in \mathcal{R}} I_{ij} \hat{S}_j \lambda_j^0. \quad (4.21)$$

Thus, when using a zero initialization $\lambda^0 = 0$, one needs one extra neighbor-to-neighbor step for initialization of d-CG. When using warm starting—i.e. using the solution of the previous outer bi-level ALADIN iteration as an initial guess $\lambda_j^0 \neq 0$—one needs one additional neighbor-to-neighbor step for computing the second term in (4.21). Note that the vectors λ_j^0 are all known locally from the previous outer bi-level ALADIN iteration. For d-ADMM such a step is not required as $\hat{s}_i$ is explicitly considered in step 1) of Algorithm 6. For warm starting, one can use λ_i^0 directly from the previous iteration.

4.3 Comparing d-ADMM and d-CG

Next, we briefly compare important properties of d-ADMM and d-CG.

4.3.1 Information exchange

The required information exchange and the locally maintained variables for d-ADMM (Algorithm 6) and d-CG (Algorithm 7) are graphically illustrated in Figure 4.5. Both algorithms require the same amount of information exchange between neighbors: whereas d-ADMM exchanges the matrix-vector product

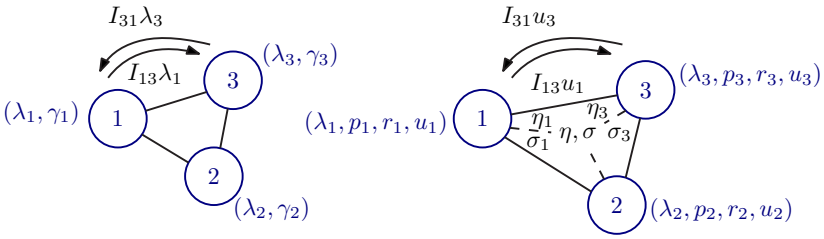


Figure 4.5: Information exchange and local variables for d-ADMM and d-CG.

Table 4.1: Properties of d-CG and d-ADMM for problem (4.12).

convergence rate	d-CG	d-ADMM
theoretical	n_c-step	linear/sublinear
practical	superlinear	linear/sublinear
tuning	no	yes
local variables	four	two

$I_{ij}\lambda_i$, d-CG exchanges the matrix-vector product $I_{ij}u_i$. A key difference in d-CG is the additionally required global communication of the two scalars η and σ per iteration. Moreover, for d-CG, one has to maintain four variables per node, whereas d-ADMM requires to maintain two variables per node.

4.3.2 Convergence properties

d-ADMM and d-CG exhibit different convergence properties. The conjugate gradient algorithm (theoretically) yields the *exact* solution in at most n_c steps [NW06] which is a very strong guarantee. However, in practice convergence is typically slower as conjugate gradient is quite sensitive to numerical round-off errors coming from finite precision arithmetic. Nonetheless one can observe superlinear convergence [BK01]. The convergence rate of ADMM is either sublinear for convex problems or linear for strongly convex problems, cf. Chapter 3. A further advantage of d-CG compared with d-ADMM is that no tuning of the step size is needed as this is done "automatically" in step 1) and

step 4) of Algorithm 7. The properties of d-CG and d-ADMM are summarized in Table 4.1.

Remark 13 (Different version of d-ADMM in [Eng+20]) In [Eng+20] we used a different version of d-ADMM. A major difference is that in the present version, redundant variables in (4.12) are eliminated and thus all f_i are strongly convex. Hence, in the present case, we obtain linear convergence (cf. [Yan+16; WO13; NOR18]) Moreover, note that the derivation of D-ADM and d-CG in [Eng+20] is different as we do not use the matrices I_{ij} there for encoding sparse communication.

4.4 Bi-level distributed ALADIN in summary

The overall *bi-level distributed ALADIN* algorithm is summarized in Algorithm 8. Observe the similarity to basic ALADIN in Algorithm 5. There are two main differences between these two algorithms: First, instead of computing full sensitivities like gradients, Hessians, Jacobians in step 1) of Algorithm 5, one computes *Schur complements* $\tilde{S}_i$ and $\tilde{s}_i$ by (4.10) reducing the amount of reuquired communication compared to Algorithm 5. Moreover, in step 2), the condensed coordination step is computed in a decentralized fashion leading to an overall *decentralized algorithm*. The residual $\|r_\lambda^k\|$ quantifies the "amount of inexactness" in the coordination step due to decentralized conjugate gradients and decentralized ADMM and will be defined in the next subsection.

4.5 Convergence analysis

Recall that d-CG and d-ADMM solve the coordination QP (4.1) with a *finite precision only*. Thus, the convergence analysis from [HFD16] can not directly be used for bi-level ALADIN. In the following we show, that the favorable convergence properties of ALADIN can be preserved if the error in the inexact solution to (4.1) and the error due to Hessian approximation decays fast enough. For doing so, we use techniques from inexact Newton methods [DES82; HFD16]. Our analysis contains the analysis of basic ALADIN as a special case. The overall proof is outlined in Figure 4.6. The main idea is to

Algorithm 8. Bi-level distributed ALADIN

Initialization: Initial guess (z^0, λ^0), parameters $\Sigma_i \succ 0, \nu, \rho > 0$.
Repeat:

1) Solve in *parallel* for all $i \in \mathcal{R}$

$$x_i^k = \underset{x_i \in X_i}{\operatorname{argmin}} \quad f_i(x_i) + (\lambda^k)^\top A_i x_i + \frac{\nu}{2} \left\| x_i - z_i^k \right\|_{\Sigma_i}^2 \tag{4.22}$$

and compute *condensed* sensitivities $\tilde{S}_i$ and $\tilde{s}_i$.

2) *Coordination Step:* Solve (4.9) centrally *or* decentrally with d-CG or d-ADMM with residuum $\|r_\lambda^k\|$ small enough such that (4.33) holds for λ^{k+1}.

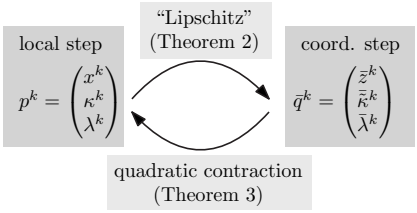
3) Set $\lambda^{k+1} \leftarrow \lambda^k$ and $z_i^{k+1} \leftarrow x_i^k + \Delta x_i^k$.

combine the property that the local problems in step 1) of ALADIN form a Lipschitz continuous map in terms of (z^k, λ^k) and step 2) can be viewed as an SQP step making results from Newton-type methods for fast local contraction applicable (cf. Section 2.1.2).

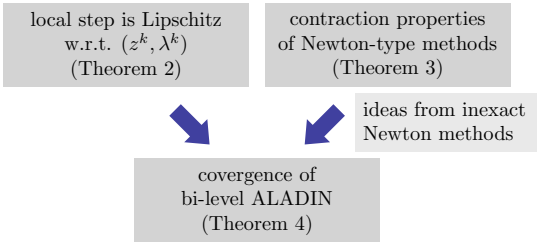
Lipschitzness of the parallel step

Let us start by analyzing the Lipschitz property of the parallel step 1) in basic ALADIN (Algorithm 5). This step is important mainly because of two reasons: It minimizes the local NLPs while at the same time staying close to the previous iterate z^k, where the distance to z^k can be controlled by ν. The hope is that the minimizer of the NLP brings us closer to a local minimizer (although this is theoretically not proven so far, but often it does in practice). The second and maybe even more important feature of this step is that it determines an active set $\mathcal{A}(x^k)$ for which the constraint linearizations of the inequality constraints are held constant in the coordination QP. This renders the coordination QP an *equality* constrained QP instead of a mixed equality-inequality constrained QP, which is substantially cheaper to solve.

Next, we use standard results from parametric programming to show that the mapping formed by step 1) of basic ALADIN is Lipschitz continuous with respect to (z^k, λ^k). We will then use this result to show show overall fast linear



(a) Variables in local/coordination step.



(b) Proof dependencies.

Figure 4.6: Outline of the bi-level ALADIN proof.

to quadratic local convergence of bi-level ALADIN. The theorem is originally from [Rob80, Thm 4.1], but here stated in form of [FI90, Thm 5.2].

Theorem 5 (Lipschitz property for parametric programming) Consider the parametric programming problem

$$\min_{x \in \mathbb{R}^n} f(x,p) \quad \text{subject to} \quad g(x,p) = 0 \quad \text{and} \quad h(x,p) \leq 0. \tag{4.23}$$

Let LICQ (Definition 2) and the conditions of Theorem 2 (SOSC) and hold at $(x^\star, \bar{p})$ with $\mu^\star > 0$ (strict complementarity), where $x^\star$ is the solution to (4.23) for a given $p = \bar{p}$. Then there exists a $\chi < \infty$ such that in a neighborhood of $\bar{p}$ it holds true that

$$\|x - x^\star\| \leq \chi \|p - \bar{p}\|. \tag{4.24}$$

By defining $p^\top := (\lambda^{k\top}, z_i^{k\top})$ it is obvious to see that the local problems in step 1) in Algorithm 5 are in form of (4.23). Thus, if we are able to ensure

that LICQ, SOSC and strict complementarity hold at $x_i^\star$, we can use (4.24) in our local convergence analysis. Hence, we have to ensure that the Hessian of the Lagrangians sufficiently positive definite to ensure that SOSC holds. This can be ensured by choosing ν and Σ_i large enough/sufficiently positive definite such that locally

$$\nabla_{xx}^2 \left(f_i(x_i^k) + \kappa_i^{k\top} h_i(x_i^k) \right) + \nu \Sigma_i \succ 0 \tag{4.25}$$

is satisfied. This corresponds to the statement in [HFD16, Lemm 3].

Coordination is a Newton-type step

Next, we analyze the progress towards a local minimizer in the coordination QP (4.1). We evaluate the first-order KKT conditions (2.2) for the coordination QP which is a linear system of equations. Then we compare this linear system to the KKT system from an Newton-type step applied to the KKT conditions of problem (2.12) to highlight their similarity. This will make convergence results from Newton-type methods (Theorem 4) applicable yielding local quadratic convergence of ALADIN.

Let us start by evaluating the KKT conditions for the coordination QP (4.1). Note that as $B_i^k \succ 0$ on the nullspace of C_i^k, the coordination QP (4.1) is strongly convex and thus the KKT conditions are necessary and sufficient for solving (2.25) (cf. Section 2.1.1). The Lagrangian to (4.1) is

$$\begin{aligned}\mathcal{L}(\Delta x, s, \tilde{\kappa}, \lambda) = \frac{1}{2}\Delta x^\top B^k \Delta x + \nabla f(x^k)^\top \Delta x + \lambda^{k\top} s + \frac{\rho}{2}\|s\|_2^2 \\ + \tilde{\kappa}^\top C^k \Delta x + \lambda^\top \left(A\left(x^k + \Delta x\right) - b - s \right),\end{aligned}$$

where $\tilde{\kappa}^\top = \left(\tilde{\kappa}_1^\top, \dots, \tilde{\kappa}_R^\top\right)$ are the Lagrange multipliers associated with (4.1b). Here we use concatenated notation for all variables, i.e. $\Delta x^\top = \left(\Delta x_1^\top, \dots, \Delta x_R^\top\right)$,

$A = (A_1, \dots, A_R)$, $B^k := \operatorname{diag}_{i\in\mathcal{R}}(B_i^k)$, $C^k := \operatorname{diag}_{i\in\mathcal{R}}(C_i^k)$ and $\nabla f(x^k)^\top = \left(\nabla f_1(x_1^k)^\top, \dots, \nabla f_R(x_R^k)^\top\right)$. Thus, the KKT conditions read

$$\begin{aligned}
\nabla_{\Delta x}\mathcal{L}(\Delta x, s, \tilde{\kappa}, \lambda) &= B^k\Delta x + \nabla f(x^k) + A^\top\lambda + C^{k\top}\tilde{\kappa} &&= 0, && \text{(4.26a)}\\
\nabla_{s}\mathcal{L}(\Delta x, s, \tilde{\kappa}, \lambda) &= \lambda^k + \rho s - \lambda &&= 0, && \text{(4.26b)}\\
\nabla_{\tilde{\kappa}}\mathcal{L}(\Delta x, s, \tilde{\kappa}, \lambda) &= C^k\Delta x &&= 0, && \text{(4.26c)}\\
\nabla_{\lambda}\mathcal{L}(\Delta x, s, \tilde{\kappa}, \lambda) &= A\left(x^k + \Delta x\right) - b - s &&= 0. && \text{(4.26d)}
\end{aligned}$$

Rearranging (4.26b) yields $s = \frac{1}{\rho}(\lambda - \lambda^k)$. Eliminating s in (4.26d), defining $\Delta\lambda := \lambda - \lambda^k$, and expanding (4.26a) with $\pm C^{k\top}\kappa$ and $\pm A^\top\tilde{\lambda}^k$ reveals that

$$\begin{aligned}
\nabla_{\Delta x}\mathcal{L}(\Delta x, s, \tilde{\kappa}, \lambda) &= B^k\Delta x + \nabla f(x^k) + A^\top\Delta\lambda + A^\top\lambda^k + C^{k\top}\Delta\kappa + C^{k\top}\kappa &&= 0,\\
\nabla_{\tilde{\kappa}}\mathcal{L}(\Delta x, s, \tilde{\kappa}, \lambda) &= C^k\Delta x &&= 0,\\
\nabla_{\lambda}\mathcal{L}(\Delta x, s, \tilde{\kappa}, \lambda) &= A\left(x^k + \Delta x\right) - b - \rho^{-1}\Delta\lambda &&= 0.
\end{aligned}$$

with $\Delta\kappa = \tilde{\kappa} - \kappa$. Thus,

$$\underbrace{\begin{pmatrix} B^k & A^\top & C^{k\top} \\ A & -\frac{1}{\rho}I & 0 \\ C^k & 0 & 0 \end{pmatrix}}_{=:M\left((x^\top,\,\kappa^\top)^\top\right)} \begin{pmatrix} \Delta x \\ \Delta\lambda \\ \Delta\kappa \end{pmatrix} = \underbrace{\begin{pmatrix} -\nabla f(x^k) - A^\top\lambda^k - C^{k\top}\kappa \\ -Ax^k + b \\ 0 \end{pmatrix}}_{=:m\left((x^\top,\,\kappa^\top,\,\lambda^\top)^\top\right)}. \tag{4.28}$$

The dependence of M on κ stems from the dependence of $\nabla^2_{xx}\mathcal{L}(x, \lambda, \kappa)$ on κ (in case of exact Hessians).

Similarity to an SQP step for problem (2.12)

The Lagrangian of the affinely-coupled separable problem (2.12) reads

$$\mathcal{L}(x, \kappa_E, \kappa_I, \lambda) = f(x) + \kappa_E^\top g(x) + \kappa_I^\top h(x) + \lambda^\top(Ax - b),$$

where $f = \sum_{i \in \mathcal{R}} f_i$, $\kappa_E^\top = (\kappa_{E1}^\top, \ldots, \kappa_{ER}^\top)$, $\kappa_I^\top = (\kappa_{I1}^\top, \ldots, \kappa_{IR}^\top)$, $g^\top = (g_1^\top, \ldots, g_R^\top)$, $h^\top = (h_i^\top, \ldots, h_R^\top)$ and $A = (A_1, \ldots, A_R)$. Then, the KKT conditions read

$$\begin{aligned} F(q^\star) &:= \nabla \mathcal{L}(x^\star, \kappa_E^\star, \kappa_I^\star, \lambda^\star) \\ &= \begin{pmatrix} \nabla f(x^\star) + \nabla g(x^\star)^\top \kappa_E^\star + \nabla h(x^\star)^\top \kappa_I^\star + A^\top \lambda^\star \\ Ax^\star - b \\ g(x^\star) \\ (h(x^\star))_{\mathcal{A}(x^\star)} \end{pmatrix} = 0, \end{aligned} \tag{4.29}$$

where $q^\top := (x^\top, \kappa^\top, \lambda^\top)$ with $\kappa^\top := (\kappa_E^\top, \kappa_I^\top)$. An exact Newton step applied to (4.29), $\nabla F(q^k) \Delta q = F(q^k)$, yields

$$\begin{aligned} &\begin{pmatrix} \nabla_{xx}^2 \mathcal{L}(x^k, \kappa_E^k, \kappa_I^k, \lambda^k) & A^\top & \nabla g(x^k)^\top & \nabla \left(h(x^k)\right)_{\mathcal{A}(x^\star)}^\top \\ A & 0 & 0 & 0 \\ \nabla g(x^k) & 0 & 0 & 0 \\ \nabla \left(h(x^k)\right)_{\mathcal{A}(x^\star)} & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta \lambda \\ \Delta \kappa_E \\ \Delta \kappa_I \end{pmatrix} \\ &\qquad = \begin{pmatrix} -\nabla f(x^k) - \nabla g(x^k)^\top \kappa_E^k - \nabla h(x^k)^\top \kappa_I^k - A^\top \lambda^k \\ -Ax^k + b \\ -g(x^k) \\ -\left(h(x^k)\right)_{\mathcal{A}(x^\star)} \end{pmatrix}. \end{aligned} \tag{4.30}$$

By comparing (4.30) with (4.28) one can see that for exact Hessians $B^k = \nabla_{xx}^2 \mathcal{L}$ and Jacobians $C^{k\top} = \left(\nabla g(x^k)^\top \ \nabla \left(h(x^k)\right)_{\mathcal{A}(x^\star)}^\top\right)$, both equations coincide apart from the term $-\frac{1}{\rho} I$. For $\rho \to \infty$ both equations coincide exactly. Hence, the coordination step of ALADIN can be interpreted as an *inexact Newton step* to (2.12). Note that Theorem 4 allows for such an "inexactness" in the compatibility condition (2.10a), but the convergence rate depends on whether this inexactness vanishes asymptotically.

Overall convergence under inexact coordination

Now let us use Theorem 4 to prove local convergence of ALADIN. For doing so, we have to check whether the conditions of Theorem 4 are satisfied. The

Lipschitz condition (2.10b) holds by assuming Lipschitzness of the gradients and Hessians/Jacobians in f, g and h in (2.12) and by assuming that LICQ and SOSC are satisfied (to exclude the case that the KKT matrix is rank-deficient at $q^\star$). For the compatibility condition (2.10a), let us assume that we use exact Hessians $B^k = \nabla^2_{xx}\mathcal{L}(x^k, \kappa^k, \lambda^k)$. Then, the compatibility condition reads (by using the left-hand sides of (4.30) and (4.28))

$$\gamma^k := \left\|(M^k(\rho^k))^{-1}(M^k(\rho^k) - \nabla F(q^k))\right\| = \left\|(M^k(\rho^k))^{-1}\right\| \left\|1/\rho^k \cdot I\right\|.$$

Since $M^k(\rho^k) \to \nabla F(q^k)$ for $\rho^k \to \infty$, $\left\|(M^k(\rho^k))^{-1}\right\|$ converges to a fixed number and $1/\rho^k$ converges to zero which means that we can make γ^k arbitrarily small (and especially smaller than one) as required by Theorem 4. Thus, the conditions of Theorem 4 are satisfied[3] and we can use the contraction inequality (2.9) in our following convergence analysis.

Apart from the inexactness introduced by the term $\frac{1}{\rho}I$ from before, we have a second kind of inexactness: the inexactness coming from an inexact step computation from the before-mentioned two inner algorithms d-ADMM and d-CG. Thus, we include this second source of inexactness in our convergence analysis. For doing so, we use ideas from inexact Newton methods [DES82]. Let us denote the primal-dual iterate after the coordination step 3) of basic ALADIN (Algorithm 5) with $q^\top = (z^\top, \tilde{\kappa}^\top, \lambda^\top)$ and let us denote an *inexact primal-dual iterate* to step 3) by $\bar{q}$.

We analyze the effect of the inexactness. The distance of the inexact iterate $\bar{q}^{k+1}$ to a local primal-dual solution $q^\star$ reads

$$\begin{aligned}\left\|\bar{q}^{k+1} - q^\star\right\| \leq \left\|\bar{q}^{k+1} - q^{k+1}\right\| + \|q^{k+1} - q^\star\| \leq \\ \left\|\bar{q}^{k+1} - q^{k+1}\right\| + \frac{\omega}{2}\|p^k - p^\star\|^2 + \gamma\left\|p^k - p^\star\right\|, \quad (4.31)\end{aligned}$$

where we expanded with $\pm q^{k+1}$, used the triangular inequality and the Newton-type contraction from Theorem 4, where we denote the primal-dual iterate after step 1) of basic ALADIN with $p^\top = (x^\top, \kappa^\top, \lambda^\top)$. In order to preserve convergence, we need to bound $\left\|\bar{q}^{k+1} - q^{k+1}\right\|$ in (4.31) representing the "inexactness"

[3] We consider local convergence here. Thus, the condition $\|q^k - q^\star\| < \frac{2(1-\gamma)}{\omega}$ has to be satisfied by a suitable primal-dual initialization.

of the solution to (4.1). In order to do so, let us define the residual to the KKT system (4.28) as

$$r_q^k := M(p^k)\,(\bar{q}^{k+1} - p^{k+1}) - m(p^k, \lambda^k). \tag{4.32}$$

Since $M(p^k)\,(q^{k+1} - p^{k+1}) - m(p^k, \lambda^k) = 0$ by definition of q^{k+1}, we have

$$\begin{aligned}\|\bar{q}^{k+1} - q^{k+1}\| &= \|(M(p^k))^{-1}\big(r_q^k + m(p^k, \lambda^k) - m(p^k, \lambda^k)\big)\| \\ &\leq \|(M(p^k))^{-1}\|\|r_q^k\|.\end{aligned}$$

Combination with (4.31) and defining $\beta := \|M(p^k)^{-1}\|$ yields

$$\left\|\bar{q}^{k+1} - q^\star\right\| \leq \beta\|r_q^k\| + \frac{\omega}{2}\|p^k - p^\star\|^2 + \gamma\left\|p^k - p^\star\right\|.$$

By assuming LICQ, strict complemntarity and SOSC we know that by Theorem 5 that the mapping formed by the local problems in ALADIN Lipschitz with constant χ. Thus, we get

$$\left\|\bar{q}^{k+1} - q^\star\right\| \leq \left(\beta\|r_q^k\| + \gamma^k\right)\chi\left\|\bar{q}^k - q^\star\right\| + \frac{\omega}{2}\chi^2\left\|\bar{q}^k - q^\star\right\|^2.$$

Enforcing a bound on the inaccuracy in the solution to the coordination QP stemming from inner algorithms

$$\|r_q^k\| \leq \eta^k\|m(p^k, \lambda^k)\| \tag{4.33}$$

yields

$$\left\|\bar{q}^{k+1} - q^\star\right\| \leq \left(\beta\,\delta\,\eta^k + \gamma^k\right)\chi\left\|\bar{q}^k - q^\star\right\| + \frac{\omega}{2}\chi^2\left\|\bar{q}^k - q^\star\right\|^2, \tag{4.34}$$

where δ is a Lipschitz constant of m.

Note that if the sequences $\{\eta^k\}$ and $\{\gamma^k\}$ are bounded by constants η, γ small enough such that $\|\bar{q}^k - q^\star\| < 2(1 - \chi\,\beta\,\delta\,\eta^k - \chi\,\gamma^k)(\omega\,\chi^2)^{-1}$ is satisfied in each iteration, linear convergence of ALADIN is guaranteed. By additionally enforcing bounds on the sequences $\{\eta^k\}$ and $\{\gamma^k\}$, i.e. $\eta^k = O(\|\bar{q}^k - q^\star\|)$ and $\gamma^k = O(\|\bar{q}^k - q^\star\|)$ we can make the convergence quadratic. Alternatively, enforcing $\eta^k, \gamma^k \to 0$ yields superlinear convergence.

We summarize our results from this section in the following Theorem.

Theorem 6 (Convergence of bi-level distributed ALADIN)
Consider bi-level distributed ALADIN (Algorithm 8) with Σ_i and ν such that (4.25) holds. Suppose Assumption 2 and the conditions of Theorem 2 hold. Let the solution to the condensed QP (4.9) satisfy (4.33) for a bounded sequence $\{\eta^k\}_{k\in\mathbb{N}} < \eta$ and let q^0 be close enough to a primal-dual minimizer $q^\star$.

Then bi-level distributed ALADIN converges to $p^\star$

- at linear rate if η and γ are small enough; and
- at quadratic rate if $\eta^k = O(\|\bar{q}^k - q^\star\|)$ and $\gamma^k = O(\|\bar{q}^k - q^\star\|)$.

Theorem 6 follows directly from inequality (4.34) and the definitions of linear and quadratic convergence (cf. Section A.1).

In basic ALADIN, the coordination problem (4.9) is solved exactly and thus $\eta^k \equiv 0$. Thus, basic ALADIN can be seen as a *special case of bi-level ALADIN*. Moreover, the convergence analysis given here can be extended to the case where the the local NLPs are solved with a finite precision only [Eng+19b] using similar techniques as we did here. We omit this extension as it adds more technical complications without adding much insight. Moreover, a BFGS-ALADIN variant we presented in [Eng+19b] is included in this framework as a particular choice of B_i^k. We do not consider this variant here.

Remark 14 (KKT matrix modifications) Note that the compatibility condition (2.10a) and the corresponding requirements on $\{\gamma^k\}$ cover a variety of modifications of the KKT matrix M—not only the modifications with respect to $\frac{1}{\rho}I$. The error due to Hessian approximations like BFGS or the error due to regularization procedures for ensuring positive definiteness of B_i^k can be further sources of inexactness, or errors because of approximations in the constraint linearizations C^k.

Remark 15 (Evaluation of the error condition (4.33)*)* In order to guarantee local convergence one has to ensure that (4.33) holds for a suitable sequence $\{\eta^k\} < \eta$. Intuitively this means, that the residual $\|r_q^k\|$ has to become smaller as ALADIN proceeds and thus the accuracy in the coordination step has to get higher as we get closer to a local minimizer as $m(p^k, \lambda^k) \to 0$ for

$q^k \to q^\star$. But evaluating $\|r_q^k\|$ at each iteration is expensive and yields additional overhead, which one would like to avoid. There is an option to overcome this limitation: instead of evaluating $\|r_q^k\|$ one can equivalently evaluate the residual in the reduced system (4.9), r_λ^k. This can be seen as follows. As we use the nullspace method, the last row of m in (4.28) is always zero by $C^k Z^k = 0$. Hence, it remains to evaluate the first and second row of (4.28). Again, by using the nullspace-parametrization $\Delta x = Z^k \Delta v$ from Section 4.1, one can *enforce* a zero residual in the first row (by multiplying with $Z^{k\top}$ from the left and exploiting $C^k Z^k = 0$)

$$\bar{B}^k \Delta v + \bar{g}^k + \bar{A}^k \bar{\lambda}^{k+1} = 0,$$

where we recall that $\bar{\lambda}^{k+1}$ denotes the inexact version of λ^{k+1}. Inserting into the second row yields the residual

$$r_\lambda^k = (\bar{A}^k (\bar{B}^k)^{-1} \bar{A} + \tfrac{1}{\rho} I)\lambda^{k+1} + \bar{A}^k (\bar{B}^k)^{-1} \bar{g} - \bar{A} v^k + \tfrac{1}{\rho}\lambda^k + b = \tilde{S}\lambda^{k+1} - \tilde{s}.$$

With that, we have $r_q^k = (0^\top \;\; 0^\top \;\; r_\lambda^{k\top})^\top$ and thus $\|r_q^k\| = \|r_\lambda^k\|$. Hence, we can use r_λ^k for evaluating (4.33) instead of r_q^k.

4.6 Comparison of variants

Next, we compare the two developed bi-level variants with basic ALADIN, ALADIN with condensing and ADMM as one of the most prominent primal-dual methods. We denote bi-level ALADIN with ADMM with b-ADMM in the following, and bi-level ALADIN with conjugate gradients with b-CG. Our evaluation mainly focuses on communication effort, coordination effort, convergence guarantees and practical convergence.

Communication

As reducing communication was one of our main motivations for developing bi-level ALADIN, let us start by analyzing the amount of communication. We analyze communication by counting the number of floating point numbers (floats) which have to be exchanged per iteration.

We begin by analyzing basic ALADIN. Here we have to communicate all components necessary to construct problem (4.1), i.e. the Hessian approximations B_i, the gradients of the local objective functions $\nabla f_i(x_i^k)$, the Jacobians C_i^k and the primal variables x_i. Note that we do not count the communication of A_i and b since they stay constant during the iterations and have to be communicated only once. The Hessian approximation B_i^k is a symmetric matrix, requiring to communicate $n_{xi}(n_{xi}+1)/2$ floats. The Jacobian C_i^k calls for communicating $(n_{gi}+n_{hi}^k)\, n_{xi}$ floats, where n_{hi}^k is the number of active inequality constraints in outer iteration k. Furthermore, the primal iterates x_i and the gradients $\nabla f_i(x_i^k)$ lead to n_{xi} additional floats from each subsystem. In total, this means that we get a forward commincation for basic ALADIN of

$$\sum_{i\in\mathcal{R}} n_{xi}(n_{xi}+1)/2 + n_{xi}(n_{gi}+n_{hi}^k) + 2n_{xi}$$

floats per ALADIN iteration if we do not exploit structural zeros in any of the before mentioned vectors/matrices. In the backward step 3) of ALADIN (Algorithm 5), z_i^{k+1} and λ^{k+1}have to be communicated leading to $(\sum_{i\in\mathcal{R}} n_{xi} + n_c)$ floats in backward communication.

For the condensed ALADIN variant, all subsystems have to communicate their reduced Schur-complement matrices $\tilde{S}_i \in \mathbb{R}^{|C(i)|\times|C(i)|}$ and vectors $\tilde{s}_i \in \mathbb{R}^{|C(i)|}$. As $\tilde{S}_i$ is symmetric, this leads to a forward communication of

$$\sum_{i\in\mathcal{R}} |C(i)|\big(|C(i)|+1\big)/2 + |C(i)|.$$

In the backward step only λ^{k+1} has to be communicated leading to n_c floats in backward communication.

For b-ADMM (Algorithm 6), the matrix vector product $I_{ij}\lambda_j^{n+1}$ has to be communicated in each inner iteration. As the total number of non-zero entries in I_{ij} is equal to the number of consensus constraints assigned to subsystem $i \in \mathcal{R}$, we have $|C(i)|$ floats for for each inner ADMM iteration. Thus, we need to communicate $\sum_{i\in\mathcal{R}} |C(i)|$ in each d-ADMM iteration on a neighbor-to-neighbor basis. This yields a total forward communication of $n^{\mathrm{AD}} \sum_{i\in\mathcal{R}} |C(i)|$ floats per outer ALADIN iteration, where n^{AD} is the number of d-ADMM iterations. As all λ_i^n are known locally already during the d-ADMM iterations,

Table 4.2: Per-step forward communication (number of floats) for different bi-level ALADIN variants and ADMM.

ALADIN variant	global	local	backward
basic	$\sum_{i\in\mathcal{R}} n_{xi}\left(\frac{n_{xi}+1}{2}+n_{gi}+n_{hi}^{k}+2\right)$	-	$\sum_{i\in\mathcal{R}} n_{xi}+n_c$
condensed	$\sum_{i\in\mathcal{R}} \frac{\lvert C(i)\rvert\left(\lvert C(i)\rvert+3\right)}{2}$	-	n_c
bi-level (CG)	$2n^{\mathrm{CG}}R$	$n^{\mathrm{CG}}\sum_{i\in\mathcal{R}}\lvert C(i)\rvert$	-
bi-level (ADMM)	-	$n^{\mathrm{AD}}\sum_{i\in\mathcal{R}}\lvert C(i)\rvert$	-
pure ADMM	-	$\sum_{i\in\mathcal{R}}\lvert C(i)\rvert$	-

no backward communication is needed. Furthermore, for d-ADMM there is no need for global communication.

Similar to d-ADMM, in d-CG the matrix-vector products $I_{ji}u_j$ have to be communicated leading to local communication of $n^{\mathrm{CG}}\sum_{i\in\mathcal{R}}|C(i)|$ floats, where n^{CG} is the number of inner d-CG iterations. Additionaly, d-CG needs the global communication of two floats per subsystem and per inner d-CG iteration leading to $2Rn^{\mathrm{CG}}$ floats. Also for d-CG no backward communication is required as λ_i is already known locally.

For ADMM directly applied to (2.12), there are variants similar to Algorithm 4 requiring the exchange of the coupling variables only [Boy+11]. This leads to a communication overhead of $\sum_{i\in\mathcal{R}}|C(i)|$ floats per ADMM iteration. From this, at first glance it seems that ADMM outperforms the ALADIN variants by far in terms of communication. However, keep in mind that due to its weak convergence rate guarantees, ADMM might need much more iterations in total to achieve the same accuacy compared in ALADIN—so for evaluating the *total amount of communication* one has to multiply these numbers by the numbers of iterations needed. We will investigate this effect numerically in Section 5.6.

Table 4.3: Bi-level ALADIN compared to distributed optimization algorithms.

	primal-dual	primal	internal dec.	ALADIN	bi-level ALADIN
decentralization	+	++	- -	-	+ (++)
convergence guarantees	conv	conv	non-conv	non-conv	non-conv
convergence speed	o	(- -), o	++	+++	++ (+++)
communication	++	++	- -	- -	+
broad applicability	++	- -	+	+	+
advanced features	+	++	- -	- -	- -

Convergence properties and rates

As shown in Theorem 6, bi-level has similar (local) convergence properties than basic ALADIN with one difference: the convergence rate here depends on the accuracy in the coordination step and on the quality of the Hessian approximations. If the accuracy increases and the approximation error decreases fast enough in terms of the distance to a local minimizer, quadratic convergence rate can be preserved.

The properties of bi-level ALADIN compared with primal, primal-dual and internal decomposition methods are summarized in Table 4.3. Here, one can see that with bi-level ALADIN we overcome the two main limitations of basic ALADIN:

- the required amount of communication,
- and the lack of decentralization.

At the same time, the favorable convergence properties of ALADIN are to a large extend preserved.

It remains to show how these methods perform in practice. We will do so in the next section with main focus on problems from power systems. A further example from distributed optimal control is given in Section 6.6.

4.7 Summary and conclusion

In this section we presented a framework called bi-level ALADIN, which is one of the first decentralized optimization algorithms for non-convex optimization with fast local convergence guarantees. In bi-level ALADIN, we use methods from nonlinear programming such as the nullspace method and the Schur-complement to substantially reduce the dimension of the coordination QP of basic ALADIN. Moreover, we showed how to exploit problem structure to solve the coordination problem in a decentralized fashion making bi-level ALADIN a decentralized algorithm. We presented two algorithms for doing so—a variant of ADMM derived similar to other variants in the literature and a decentralized variant of the conjugate gradient method, which is to the best of our knowledge novel. The versions of d-ADMM and d-CG given here are improved versions of [Eng+20], where ADMM is guaranteed to convergence linearly due to a strongly convex formulation of the coordination QP. Moreover, the here used variant of the conjugate gradient method omits the precomputing step from [Eng+20] leading to a substantial improvement in local communication for d-CG. Additionally, we showed that the fast local convergence properties of ALADIN are preserved in bi-level ALADIN, if the error in the coordination step decays fast enough.

5 Application to Power Systems

This chapter evaluates the performance of ALADIN, bi-level ALADIN and ADMM on Optimal Power Flow (OPF) problems—an important problem class from power systems. We start with main motivations for using distributed optimization methods for OPF.

We consider the AC power system model from [GS94; Sch17; FR16; FGM18]. The problem formulation for distributed OPF is from [Eng+19b; Eng+17]. The numerical results of basic ALADIN and ADMM are similar to [Eng+19b; Eng+20]. The analysis of the role of a feasible initial point in Section 5.4.3 is from [EF18]. The analysis of bi-level ALADIN for OPF in Section 5.5 is mainly from [Eng+20]. Parts of the convergence analysis of d-ADMM and the numerical communication analysis are new in this thesis.

5.1 How to operate power systems?

The main purpose of power power systems is to deliver power from generators to power consumers. Thereby, the consumer's demand should be satisfied at any time and to an arbitrary amount. In a classical setting, there are no storage elements. This implies that the generated power has to match the power demand exactly in each step in time. Although power demands can typically not be influenced, there is a certain degree of freedom in power generation. Namely, one can decide which generator takes which share of total active and reactive power demand satisfaction. This can be exploited for a cost-effective operation.

Engineering constraints also have to be considered. Transmission lines for example have thermal limits allowing them to carry a limited current only. Consumer devices and grid components are only guaranteed to work properly in specified ranges around the nominal voltage. Having this in mind, one can

ask the following important question: how to choose the generator inputs such that

- all power *demands* are *satisfied*;
- all *technical limitations* are met;
- and in a *cost-effective* fashion?

One of the most promising ways of approaching this question is to encode the above question in an optimization problem called the Optimal Power Flow problem. Optimal power flow computes cost-optimal generator set points while satisfying power demands and technical limitations. We give a brief introduction to OPF in Appendix B.

New challenges in today's power system

For formulating an OPF problem, a grid model is required. Grid models, however, contain sensitive data. This includes the model data itself but also consumption data potentially revealing sensitive information about consumer behavior. Moreover, power system are often required to be $(N-1)$-secure, that is, a single point of failure is not acceptable due to data aggregation and the existence of a single point of failure. Moreover, in many countries, there are many grid operators which are cooperatively responsible for operating the grid in an economically optimal and safe fashion. In Germany for example, there are four transmission system operators and more than 800 distribution system operators [Bun19], which have to coordinate in some way. The map of the German distribution grid operators, Figure 5.1, graphically illustrates the dimension of this problem. Here, the question arises how to coordinate these grid operators.

Distributed and especially decentralized optimization algorithms seems to be an excellent fit for addressing the above challenges. Distributed optimization offers systematic coordination while reducing complexity by shifting computation overhead to the subsystems (grid operators). At the same time the information exchange and centralized coordination are limited.

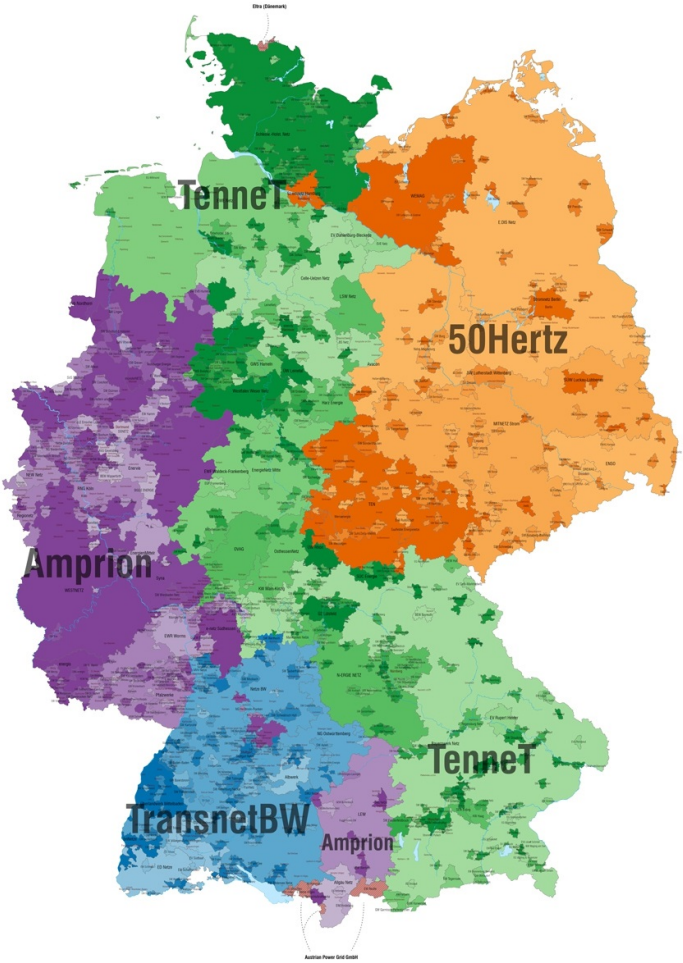


Figure 5.1: Distribution grid operators in Germany [ene20].

5.2 Distributed optimal power flow

In order to apply distributed and decentralized optimization algorithms such as ADMM and (bi-level) ALADIN, we have to express the OPF problem mathematically in structured form (1.2),

$$\min_x \sum_{i\in\mathcal{R}} f_i(x_i) \quad \text{subject to} \quad x_i \in \mathcal{X}_i, \;\; i \in \mathcal{R} \quad \text{and} \quad \sum_{i\in\mathcal{R}} A_i x_i = b. \tag{5.1}$$

Here, the set $\mathcal{R}$ represents the set of regions or system operators, f_i encodes the cost of power generation in each region, and $\mathcal{X}_i$ captures the grid model and technical limitations in each region.

Due to renewable generation, today electrical grids are often operated close to their capacity limits. Thus, one has to consider grid models capturing effects which occur in this case such as the effect of reactive power and voltage limits. The most common model considering these effects is the so-called AC model, which we recall in Appendix B. Therein, we also show how to formulate the AC model in form of (5.1). Solving (5.1) via distributed or decentralized algorithms is called distributed optimal power flow.

5.3 Literature review

Next, we provide an overview on main lines of research for distributed OPF. In Chapter 3 we have seen, that most distributed and decentralized optimization algorithms are designed for *convex problems*. Moreover, even if they can handle non-convexity, this is typically in the objective function but not in the constraints. Unfortunately, in AC optimal power flow problems we have *non-convex* constraints (cf. Appendix B). To cope with this issue, one can identify four main lines of research in the literature:

- Early works propose *internal decomposition* methods mainly focusing on decomposing the KKT system in Newton's method or in interior point methods [Com92].
- The second line of research staring in the 90s of the last century applies *primal-dual* convex optimization algorithms to OPF [Ers14a; Bal+99;

HPK02; SPG13]. Although they work well in many cases, convergence can not be guaranteed and their convergence rate is often slow as we will see later in this section.

- A third line of research simplifies the problem constraints obtaining convex problems for which primal-dual distributed algorithms exhibit convergence guarantees. *Approximations* like DC optimal power flow or semidefinite *relaxations* belong to this class [Low14a; DZG13a; CA98].
- As indicated in Chapter 3, few *distributed algorithms* with convergence guarantees *for non-convex problems* exist and are applied to OPF—two of them are ALADIN and bi-level ALADIN [Eng+19b; Eng+20], and a third one is an alternating trust-region method [HJ17].

Internal decomposition methods

Let us start with internal decomposition methods as they were among the first decomposition methods applied to power flow and also to optimal power flow problems. Note that as outline in section Chapter 3, these methods were mainly designed for computational speedup coming from limited computation power at that time. One of the first works [FP78] considers a combination of clustering techniques and block elimination in the KKT system. This line of research was extensively investigated in the 1980s, e.g. for power system state estimation [WHB81] even in asynchronous schemes [TPG83]. Advanced structure-exploiting factorization techniques are used in [VNM83; BA86; SAY87; ETA90; OKS90; SVN93], where most of these works are summarized in the technical report [Com92]. Some of these block-elimination techniques have a similar flavor as the reduction methods we used in Section 4.1 for bi-level ALADIN, especially the works [FP78; SAY87]. Although these methods are transferable to OPF, in their original form they typically address power flow or state-estimation problems. More recent examples for internal decomposition in interior point methods can be found in [Yan+11] and [LT18], where in [LT18] the authors yield very promising numerical results also for large-scale systems. Note that all of the before-mentioned methods still require central coordination for solving the reduced KKT system and also for other operations such as barrier-parameter updates. Thus, these methods offer only a limited degree of distribution.

A second branch works on decomposing the KKT conditions instead of the decomposing the KKT system. The main idea is to hold all variables which are not involved in the corresponding subsystem fixed and to solve the resulting simplified subproblems independently from each other. The method doing so is called Optimality Condition Decomposition (OCD), first proposed in [CNP02; Con+06] and further investigated for OPF in [NPC03; HA09]. In [CNP02], the convergence rate of OCD is shown to be linear under certain technical conditions. One weakness of the above methods is—although they might work well in certain practical problems—that the convergence of these methods seems not to be fully analyzed at present. In [CNP02], a sufficient condition for convergence of OCD to a first-order stationary point is developed. However, to the best of our knowledge, it remains unclear whether this condition holds for arbitrary OPF problems [Ers14a]. A similar approach based on decomposing the KKT conditions without tuning parameters can be found in [BB06].

Remark 16 (Relation to classical Gauss-Seidel methods) Note that also the classical Gauss-Seidel method for power flow problems [HO93; Glo11] can be considered as a nodally "distributed algorithm". Therein, each node computes individual voltage updates based on the current iterates of neighbored nodes. However, as the non-convexity of the power flow equations suggests, also the classical literture indicates that the Gauss-Seidel method often diverges [Glo11, Chap. 6.5].

Primal-dual algorithms

We continue with (convex) primal-dual optimization algorithms based on augmented Lagragians directly applied to the non-convex AC OPF. Early primal-dual methods use dual decomposition [AQC99], ADMM [WSL01], and related augmented Lagrangian methods such as the auxiliary problem principle (APP) [Bal+99; HPK02; AKA07] for solving the OPF problem in a distributed fashion. APP is evaluated against other augmented Lagrangian methods like ADMM and a Predictor-Corrector Proximal Multiplier Method in [KB00]—all of them showing a comparable performance. This line of research continues until today—especially ADMM [Ers14b], with recently refined parameter-update schemes in [Ers15]. Although ADMM converges to modest accuracy for medium-sized grids (similar to many other augmented Lagrangian methods), recent attempts on large-scale grids indicate that convergence becomes

increasingly difficult for with increasng problems size and might work only with special partitioning techniques [GHT17]. Moreover, all of these methods are in general not guaranteed to converge and divergence seems to occur in practice [Chr+17a]. If they converge, they achieve at maximum a linear convergence rate. But the convergence modulus can be quite large as we shall see later. Moreover, if an increasing sequence of the penalty parameter in combination with a feasible initialization is used (as done for example in [GHT17]), one can show that these methods converge to a feasible but not necessarily optimal point (cf. Section 5.4.3). One strength of augmented-Lagrangian and similar methods is that they seem to converge quite stable in terms of converging to modest accuracy for OPF—at least up to medium-sized grids. Recently, an augmented-Lagrangian based scheme for radial grids with guarantees for the AC model is presented in [Liu+17].

Convex relaxations and approximations

A different approach uses convex *relaxations* and *approximations* of the power flow equations yielding convex problems for which the before presented algorithms are—in contrast to the full AC-OPF problem—guaranteed to converge. A classical approach for doing so is based on the DC power flow approximation [GS94] to which augmented Lagrangian-based approaches have been applied in [LHA94; CA98; BB03; Fei+15]. The advantage of this approach is that classical augmented Lagrangian methods converge with guarantees and with modest communication requirements for the relaxed problems. However, important physical quantities such as voltage magnitudes and the reactive power are not considered in the DC model making it non-applicable for many practical applications. Especially for distribution grids—which is one of the most important field of application for modern OPF methods—the accuracy of the DC model can be poor [BZ16; Chr+17a]. Recent approaches combining the DC model with modern distributed algorithms such as ALADIN can be found in [Jia+20; Jia+19]. Another approach for convex modeling of radial grids is the so-called LinDistFlow model [BW89] which is based on the assumption of a flat voltage profile. The resulting convex OPF problem is solved via ADMM in [ŠBC14]. However, in view of the large voltage deviations in distribution grids cause by renewable power injection, this assumptions hardly holds in many practical situations.

Table 5.1: Distributed optimization for optimal power flow—state-of-the-art.

		assumptions	guarantees	pros/cons	representative works
this thesis	ALADIN	good initialization	yes	+ performance + accuracy - communication - coordination	[Eng+17; Eng+19b; MSL19], (ALADIN-DC: [Jia+20; Jia+19])
	ADMM	convexity, (feasible initialization, large ρ)	no	+ coordination + communication - performance - accuracy	[Ers14a; Ers15; KB00; Bal+99; DZG13a; LC13]
literature	ADMM-DC	DC power flow	yes	+ simple - no voltages/ reactive power	[LHA94; CA98; BB03; Fei+15]
	ADMM-SDP	radial grid, phase shifters	(yes)	+ alg. guarantees + elegant theory - assumptions - problem inflation	[Bai+08; LL12; Mol+13; Low14a; DZG13a; Low14b; PL17; Chr+17a]
	OCD	technical condition	unclear	+ decentralization + communication - scalability, guarantees?	[NPC03; HA09; HKW15]
	internal decomposition	-	from "host" algorithm	+ convergence of host algorithm + scalability - coordination - decentralization	[FP78; TPG83; VNM83; BA86; SAY87; ETA90; OKS90; SVN93; Com92], [LT18; KFS18; Yan+11]
	alternating trust region		yes	+ high accuracy - tested for small grids only	[HJ17]
	"augmented Lagrangain-like" algorithms (APP, PCPM)	convexity	no	+ coordination + communication - performance - accuracy	[Bal+99; KB00; HPK02]

Instead of the DC approximation, a related line of research employs convex *relaxations* of the power flow equations via Semi-Definite Programming (SDP). Therein, the idea is to lift the OPF problem to a higher dimensional space in which it becomes convex if a certain rank constraint is dropped [Bai+08; Mol+13]. This relaxed and inflated problem is then solved by a distributed or decentralized convex optimization algorithm obtaining convergence guarantees [DZG13b; PL17; Chr+17b]. The crux in this approach is that the solution of the relaxed problem might not satisfy the before-mentioned rank condition and thus the solution of the original OPF might not be recoverable from the relaxed problem. Under certain assumptions on the technical equipment such as small transformer resistances or a radial grid topology, e.g. radial grids [LL12; Low14b; Low14a] this "back-mapping" is guaranteed to exist, although these assumptions pose severe limitations on the class of problems to which this approach is applicable. However, there is an ongoing debate whether the assumptions made are realistic [Chr+17a] and, moreover, one can show that the recovering the solution from an SDP relaxation might fail even for very small systems [KDS16]. Furthermore, the number of decision variables when using SDPs is squared leading to very large SDPs (especially for large systems) which can lead to tractability issues [Cap16].

Recent non-convex distributed algorithms

Recently, new algorithms designed for non-convex distributed nonlinear programming have been proposed which are able to handle the AC model directly and with guarantees without relying on relaxations or approximations. One of them is based on an alternating trust-region approach using a decentralized variant of conjugate gradients and alternating projection methods with convergence guarantees at linear rate for linearly-constrained non-convex problems [HJ17]. The approach was successfully applied to small OPF problems ranging from 9 to 56 buses. A second approach is based on ALADIN which we will present in this chapter [Eng+19b].

An overview of the previously outlined lines of research with main strengths and weaknesses of each line is given in Table 5.1. For more detailed overviews on general and distributed methods for OPF we refer to [Mol+17; Cap16]. Note that we consider ADMM and ALADIN to be in "this thesis" in Table 5.1

Table 5.2: Grid partitioning (excluding auxiliary buses).

Test Case	$\lvert\mathcal{A}\rvert$	Regions $\mathcal{N}_i$
5	4	{1, 5}, {2, 3}, {4}
118	13	{1–32, 113–115, 117}, {33–67}, {68–81, 116, 118}, {82–112}

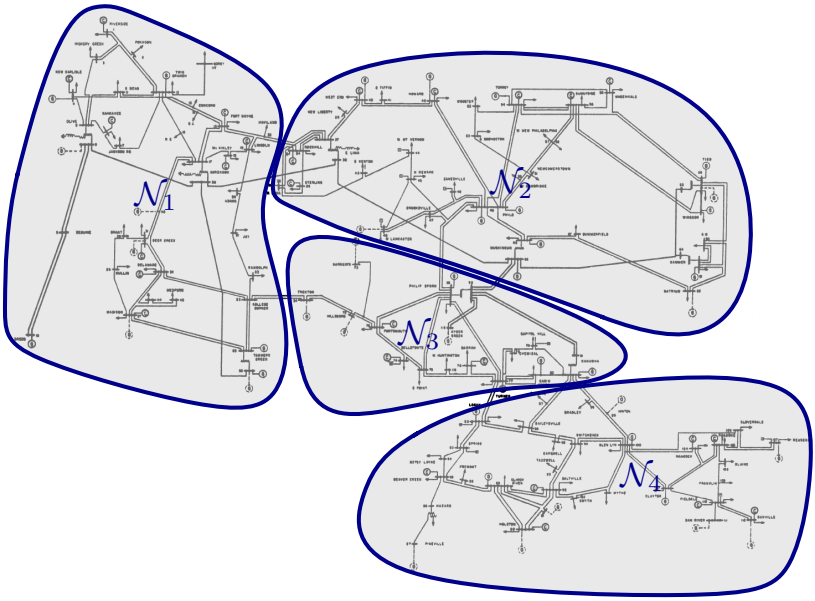
in the sense that we compare the performance of these two algorithms in this work.

Remark 17 (Related OPF problems) Considering problems with integer variables such as certain versions of the reactive power dispatch problem is beyond the scope of the present work. For first distributed approaches in this direction we refer to [Mur+18; Mur+19]. Moreover, note that although our approaches is also applicable to *temporal* decomposition, we focus on *spatial* decomposition here. Examples for temporal decomposition in OPF with storage or generator ramp constraints can be found for example in [KFS18; AC00; XS01; CCD05]. Therein, similar to internal decomposition methods, the main goal is computational tractability.

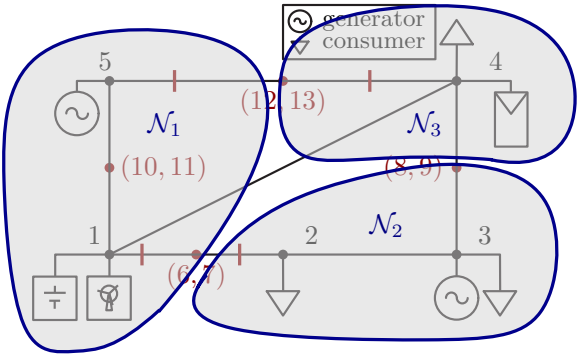
Note that these problem formulations have strong interconnections to the theory of *economic model predictive control* [Fau+18] originally developed for the optimal control of time-invariant dynamical systems. First attempts connecting these two worlds and transferring latest methodological developments from this field to power systems are proposed in [FGM18; FE19; Köh+17].

5.4 ADMM for Optimal Power Flow

Now, let us apply distributed optimization algorithms from Section 2.2 to OPF. We start with ADMM for two reasons: first, as outlined in Section 5.3, ADMM is one of the most prominent methods for distributed optimization in general and particularly for distributed OPF [Ers14b; KB00; GHT17]. Secondly, we will use ADMM as a benchmark algorithm for evaluating the performance of ALADIN and the bi-level ALADIN scheme.



(a) IEEE 118-bus system.



(b) Modified 5-bus system from [LB10].

Figure 5.2: Numerical examples.

Throughout this chapter, we use a 5-bus system from [LB10] as a tutorial example and the IEEE 118-bus as a more realistic test system as numerical examples. Both systems are shown in Figure 5.2. The corresponding partitioning are given in Table 5.2. In al cases, we tune ADMM for best performance. Note that the 5-bus system is particularly designed to be hard: we chose a partitioning in which a large amount of power has to be exchanged between region one (as mainly generating region) and the other two regions (as mainly demanding regions). Moreover, we put line limits particularly on lines connecting regions which will lead to difficulties—especially for ADMM as we shall see later. Results for ALADIN and ADMM for further benchmark systems up to 300 buses can be found in [Eng+19b].

5.4.1 Typical numerical results

Next, we show typical numerical results when applying ADMM to OPF problems. Figure 5.3 shows the convergence behavior of ADMM for the 5-bus example from Figure 5.2b for three different values of the tuning parameter $\rho^{\mathrm{ADM}} \in \{10^2, 10^3, 10^4\}$ neglecting line limits. The upper subfigures depict the values of all active power injections p^g and reactive power injections q^g over the iteration index $k \in \mathbb{N}$.[1] To characterize convergence, we consider three different metrics: the consensus violation $\|Ax^k\|_\infty$ at the current iterate x^k, characterizing the maximum mismatch of active/reactive power or voltage at coupling nodes; the distance to the minimizer $x^\star$, $\|x^k - x^\star\|_\infty$;[2] the distance of the current objective function value $f^k := f(x^k)$ to a local minimum $f^\star := f(x^\star)$, and $r^k := \nabla f(x^k) + \sum_{j\in\mathcal{R}} \gamma_i^{k\top} \nabla g_i(x^k) + \sum_{j\in\mathcal{R}} \rho_i^{k\top} \nabla h_i(x^k) + \lambda^{k\top}(\sum_{i\in\mathcal{R}} A_i x_i^k - b)$ for (2.12) measuring the degree of stationarity (dual feasibility) in the KKT conditions (2.2a). Note that we use the infinity norm $\|\cdot\|_\infty$ here for scale-independence.

One can see that with proper tuning of ρ^{ADM}, ADMM is able to reach a medium accuracy of $10^{-2} \ldots 10^{-3}$ within about 100 iterations. This is typical for ADMM and also observed in many other applications [Boy+11].

[1] We use the standard per-unit (p.u.) normalization system here, cf. [GS94] for details.
[2] We compute the reference solution $x^\star$ via the centralized interior-point solver IPOPT.

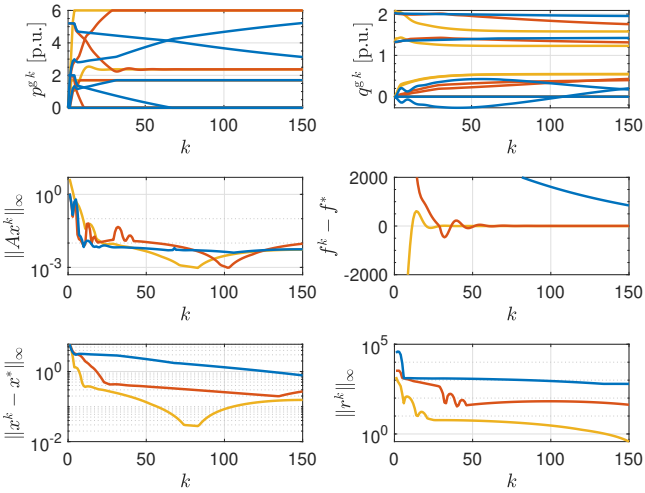


Figure 5.3: ADMM for the 5-bus system for $\rho^{\text{ADM}} \in \{10^2, 10^3, 10^4\}$.

Remark 18 (Multiple local minimizers) Note that as OPF is non-convex, there are possibly multiple local minimizers [Buk+13; LW15]. However, practical experience shows that in most cases there is only one technically meaningful high-voltage solution and local solvers converge to that solution if initialized properly [Cap16; Mom+97]. Here, all algorithms converged to the same local minimum for the standard initialization (every value zero except for the voltage magnitude which is initialized with 1, flat start). The theoretical foundations of existence and uniqueness of solutions to the power flow equations is subject to ongoing and future work [CS19; Sim18; MH19].

Scaling-up to 118 Buses

Scaling up to large grids is often hard. Different numerical issues are more likely to occur with a higher dimension, for example linearly-dependent constraint linearizations and finding an initial point close to a local minimizer might also be harder. Figure 5.4 shows the numerical performance of ADMM

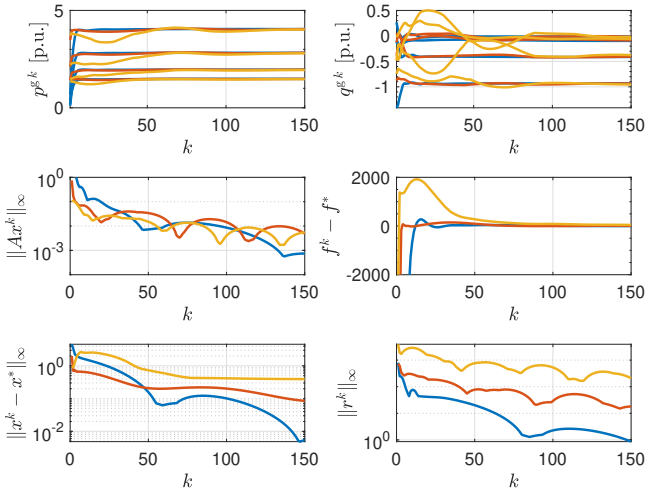


Figure 5.4: ADMM for the IEEE 118-bus system with $\rho^{\mathrm{ADM}} \in \{10^2, 10^3, 10^4\}$, and p_k^{g} at nodes $k \in \{10, 25, 26, 65\}$.

for the IEEE 118-bus system from Figure 5.2a. One can see that ADMM needs about 150 iteration to converge to a modest accuracy of 10^{-2} in the distance to the local minimizer, which is round about twice as much compared to the 5-bus system.

5.4.2 The role of special constraints

ADMM works quite well im many settings such as for the 5-bus or 118-bus OPF problem. However, next we show a practical example, where the convergence to the same level of accuracy is at least a factor of 10 larger compared with the before mentioned cases solely by adding line limits to the OPF problem. Note that this modification does neither change the problem class (non-convex NLP), nor the problem size. The line limits are indicated as red bars in Figure 5.2.

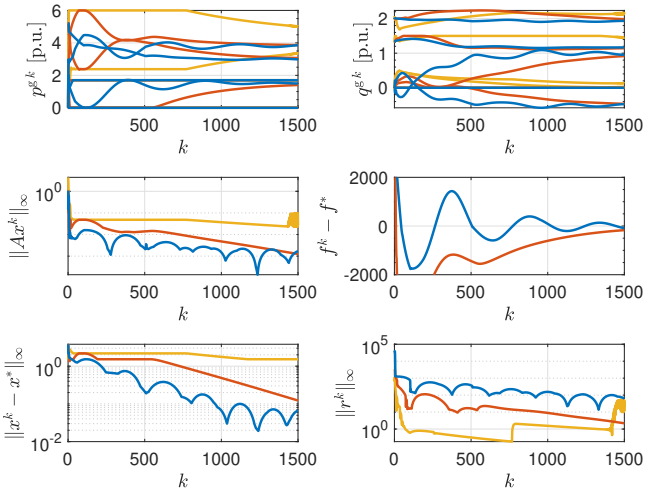


Figure 5.5: ADMM for the 5-bus system for $\rho^{\text{ADM}} \in \{10^2, 10^3, 10^4\}$ and active line limits.

The numerical results are shown in Figure 5.5. Here, ADMM needs about 1.500 iterations to reach the same level of accuracy of around 10^{-2}.

5.4.3 The role of a feasible initialization

Recent works applying ADMM to OPF problems suggest using increasing sequences for the penalty parameter ρ^{ADM} [Ers15]. In [GHT17] this rule is combine with a feasible initial point for large test systems. Next we investigate the practical and theoretical implications of these modifications.

Figure 5.6 shows the convergence behavior of ADMM for a very large value of $\rho^{\text{ADM}} = 10^9$ with and without initialization at a feasible initialization. One can see that with a feasible initialization, ADMM stays at the initial point which is feasible, but the gap of the cost to the optimal one $f^k - f^\star$ is large and does not decay to zero. This means that there is insufficient progress in terms of optimality. This can be explained by the role of the penalization parameter in

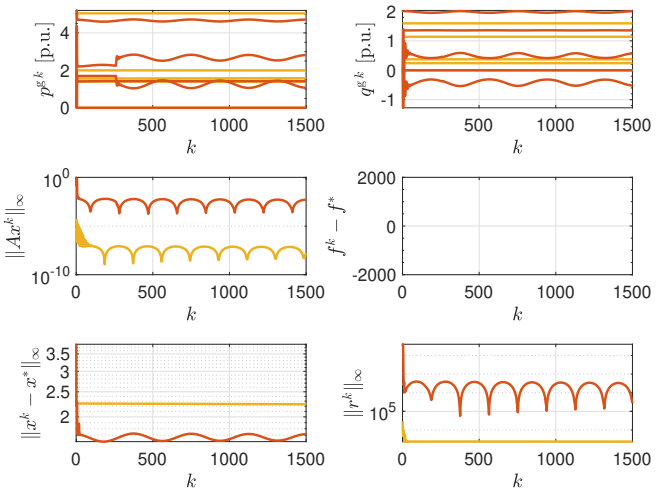


Figure 5.6: ADMM for the 5-bus system and $\rho^{\text{ADM}} = 10^9$, active line limits initialized with (yellow) and without (red) a feasible initial point.

ADMM (Algorithm 4): in case ρ^{ADM} is large, the constraints g_i and h_i are satisfied and as A_i has full row-rank, ALADIN will return $x_i^{k+1} = z_i^k$ in step 1) forced by the quadratic penalization term $\frac{\rho^{\text{ADM}}}{2}\|A_i(x_i - z_i^k)\|_2^2$. One can think of the following two steps 2) and 3) as a kind of averaging step of the coupling variables, which will also not change if the variables x_i^k do not change. Hence, ADMM gets stuck at a feasible point in this case.

Mathematical analysis

Next, we analyze this effect also mathematically for ADMM in case of $b = 0$, which we have for OPF (cf. Appendix B). Note that in the analysis we use ρ instead of ρ^{ADM} for simplified notation.

Proposition 1 (Feasibility and $\rho \to \infty$ implies $x_i^{k+1} - x_i^k \in \text{null}(A_i)$) Consider the application of ADMM (Algorithm 4) to problem (2.12) for $b = 0$. Suppose

that, for all $k \in \mathbb{N}$, the local problems in step 1) have unique minimizers x_i^k fulfilling the LICQ. For $\tilde{k} \in \mathbb{N}$, let $\lambda_i^{\tilde{k}}$ be bounded and $z^{\tilde{k}} \in \bigcup_{i\in\mathcal{R}} \mathcal{X}_i \cap C$. Then, the ADMM iterates satisfy $x_i^{\tilde{k}+1} = x_i^{\tilde{k}}$

Proof. The proof is divided into four steps. Steps 1)-3) establish technical properties used to derive the above assertion in Step 4).

Step 1). At iteration $\tilde{k}$, the local steps of ADMM are

$$x_i^{\tilde{k}}(\rho) = \underset{x_i\in\mathcal{X}_i}{\operatorname{argmin}}\ f_i(x_i) + \left(\lambda_i^{\tilde{k}}\right)^\top A_i x_i + \frac{\rho}{2}\left\|A_i\left(x_i - z_i^{\tilde{k}}\right)\right\|_2^2. \tag{5.2}$$

Now, by assumption, all f_i are twice continuously differentiable (hence bounded on $\mathcal{X}_i$), $\lambda_i^{\tilde{k}}$ is bounded and all $z_i^{\tilde{k}} \in \mathcal{X}_i$. Thus, for all $i \in \mathcal{R}$, $\lim_{\rho\to\infty} x_i^{\tilde{k}}(\rho) = z_i^{\tilde{k}} + v_i^{\tilde{k}}$ with $v_i^{\tilde{k}} \in \operatorname{null}(A_i)$.

Step 2).The first-order stationarity condition of (5.2) can be written as

$$-\nabla f_i(x_i^{\tilde{k}}) - \gamma_i^{\tilde{k}\top}\nabla h_i(x_i^{\tilde{k}}) = A_i^\top \lambda_i^{\tilde{k}} + \rho A_i^\top A_i\left(x_i^{\tilde{k}} - z_i^{\tilde{k}}\right), \tag{5.3}$$

where $\gamma_i^{\tilde{k}\top}$ is the multiplier associated to h_i. Multiplying the multiplier update formula from step 3) with $A_i^\top$ from the left we obtain $A_i^\top\lambda_i^{k+1} = A_i^\top\lambda_i^k + \rho A_i^\top A_i(x_i^k - z_i^k)$. Combined with (5.3) this yields $A_i^\top\lambda_i^{\tilde{k}+1} = -\nabla f(x_i^{\tilde{k}}) - \gamma^{\tilde{k}\top}\nabla h_i(x_i^{\tilde{k}})$. By differentiability of f_i and h_i and regularity of $x_i^{\tilde{k}}$ this implies boundedness of $A_i^\top\lambda_i^{\tilde{k}+1}$.

Step 3). Next, we show by contradiction that $\Delta x_i^{\tilde{k}} \in \operatorname{null}(A_i)$ for all $i \in \mathcal{R}$ and $\rho \to \infty$. By expressing z_i as $z_i = x_i^{\tilde{k}} + \Delta x_i$ in step 2) of ADMM, the update reads

$$\min_{\Delta x} \sum_{i\in\mathcal{R}} \frac{\rho}{2}\Delta x_i^\top A_i^\top A_i \Delta x_i - \lambda_i^{\tilde{k}+1\top} A_i \Delta x_i \quad \text{s.t.} \quad \sum_{i\in\mathcal{R}} A_i(x_i^{\tilde{k}} + \Delta x_i) = 0. \tag{5.4}$$

Observe that any $\Delta x_i^{\tilde{k}} \in \operatorname{null}(A_i)$ is a feasible point to (5.4) as $\sum_{i\in\mathcal{R}} A_i x_i^{\tilde{k}} = 0$ by step 1) of this proof and by feasibility of $z_i^{\tilde{k}}$. Consider a feasible candidate solution $\Delta x_i \notin \operatorname{null}(A_i)$ for which $\sum_{i\in\mathcal{R}} A_i(x_i^{\tilde{k}} + \Delta x_i) = 0$. Clearly,

$\lambda_i^{\tilde{k}+1\top} A_i \Delta x_i(\rho)$ will be bounded. Hence for a sufficiently large value of ρ, the objective of (5.4) will be positive. However, for any $\Delta x_i \in \text{null}(A_i)$ the objective of (5.4) is zero, which contradicts optimality of the candidate solution $\Delta x_i \notin \text{null}(A_i)$. Hence, choosing ρ sufficiently large ensures that any minimizer of (5.4) lies in $\text{null}(A_i)$.

Step 4). It remains to show $x_i^{\tilde{k}+1} = x_i^{\tilde{k}}$. Recall that we used $z^{\tilde{k}+1} = x^{\tilde{k}} + \Delta x^{\tilde{k}}$ in the previous step. Given Steps 1-3) this yields $z^{\tilde{k}+1} = z^{\tilde{k}} + v^{\tilde{k}} + \Delta x^{\tilde{k}}$ and hence

$$\left\| A_i \left(x_i - z_i^{\tilde{k}+1} \right) \right\|_2^2 = \left\| A_i \left(x_i - z_i^{\tilde{k}} + v_i^{\tilde{k}} + \Delta x_i^{\tilde{k}} \right) \right\|_2^2 = \left\| A_i \left(x_i - z_i^{\tilde{k}} \right) \right\|_2^2 .$$

Observe that this implies that, for $\rho \to \infty$, problem (5.2) does not change from step $\tilde{k}$ to $\tilde{k}+1$ by step 3) of Algorithm 4 as $\lambda^{\tilde{k}+1} = \lambda^{\tilde{k}}$ since by step 3) of this proof also $x_i^{k+1} - z_i^k$ lies in the nullspace of A_i. This proves the assertion. □

The above proposition shows that the problems in step 1) of Algorithm 4 and also λ_i^k do not change for subsequent iterates, once ADMM is at a feasible point together with $\rho \to \infty$. With the assumption that the local solvers are deterministic, i.e. they yield the same solution for same problem data this implies that ADMM stays at this feasible point, cf. Corollary 1 in [EF18].

Termination by consensus violation and alternating projections

A second important insight is that a small consensus violation $\|Ax^k\|$ does not necessarily imply that the algorithm is close to a local minimizer. Otherwise, ADMM would terminate immediately in case of a feasible initial guess and a high penalization parameter. It tells something of the feasibility of the current step, but the objective function value f^k might be far from being optimal. This is especially relevant here as $\|Ax^k\|$ is sometimes used as termination criterion for ADMM for OPF in the literature [Ers15; Ers14a]. In Appendix C we provide two additional analytical examples illustrating the above behavior.

In the opposite case of an infeasible initial guess in combination with a very high penalization, ADMM starts projecting the iterates x^k and z^k back and forth between the consensus constraint set $\mathcal{C}$ and the local nonlinear constraints (power flow equations and bounds) $\mathcal{X}_i$, cf. Algorithm 4. This can for example

Table 5.3: ALADIN parameters

	ν^0	r_ν	$\bar{\nu}$	ρ^0	r_ρ	$\bar{\rho}$
5-bus	10^3	1.1	10^8	10^4	2	$2 \cdot 10^6$
118-bus	10^2	1.1	10^8	10^3	2	$2 \cdot 10^6$

be seen by the large values of $\|Ax^k\|_\infty$ in Figure 5.6. This is similar to the examples for alternating projections from Section 2.3.1.

5.5 ALADIN for Optimal Power Flow

Next, we investigate the convergence behavior of ALADIN (Algorithm 5) and bi-level ALADIN (Algorithm 8) for OPF. In all cases, we initialize ALADIN with a flat start[3], we use the regularization technique from [Eng20a] and ALADIN parameters given in Table 5.3. Moreover, we use diagonal scaling matrices Σ with entries of 10^2 for voltage angles and magnitudes and entries 1 for active and reactive power injection.

Figure 5.7 shows the convergence of ALADIN for the 5-bus system with and without line limits, and for the IEEE 118-bus system. One can see that for all cases, ALADIN converges quite fast and almost independently of the problem size and the considered constraints (with/without line limits) in less than 20 iterations to a very high accuracy in all indicators. Observe that here we achieve accuracies of about 10^{-10} whereas for ADMM we achieved $10^{-2} \dots 10^{-3}$. This indicates that ALADIN is less scale-dependent and problem dependent compared with ADMM—likely a consequence of the relationship to SQP methods and because of constraint information in the coordination step. Moreover, note that in all cases we use a flat-start initialization—hence ALADIN does not rely on any sort of feasible initial guess. The large improvements in the accuracy of ALADIN compared with ADMM can especially be observed in the convergence of reactive power injections q^{g}: whereas after 15 iterations they are more or less converged for ALADIN in the 5-bus system (Figure 5.7

[3] In the OPF context, a flat start is the initial guess where the voltage magnitudes are initialized with 1 p.u. and all other values with 0.

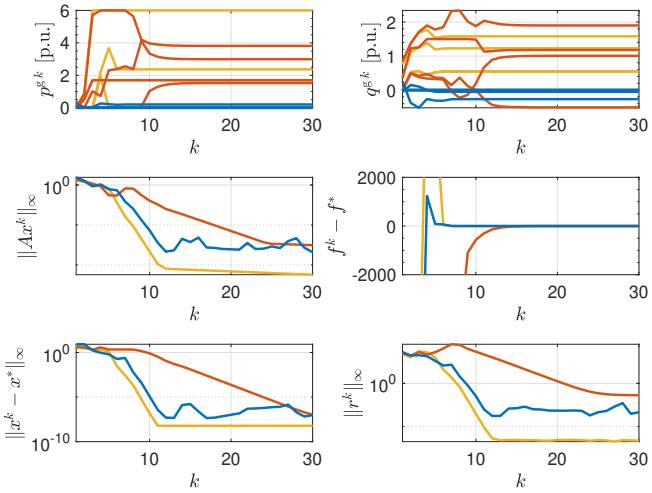


Figure 5.7: ALADIN for the 5-bus system without line limits (yellow), with line limits (red) and for the 118-bus system (blue) with flat start.

red lines), for ADMM even after 1.500 itertions there seem to be significant changes regardless of the parameter ρ^{ADM}, cf. Figure 5.5.

5.5.1 Bi-level distributed ALADIN

Next, we investigate convergence of bi-level ALADIN for the 118-bus OPF case. For b-CG we use a fixed number of 70 inner iterations and for b-ADMM we use 60, 100 and 200 inner iterations. Both variants require a minimum number of inner iterations (70 for b-CG and 60 for b-ADMM) to converge—i.e. to fulfill the accuracy requirement from inequality (4.33). We use $\rho_{\text{inn}}^{\text{ADM}} = 10^{-2}$ as the optimal penalization parameter for inner ADMM. For both inner algorithms, we use warm-starting. Pure ADMM uses $\rho^{\text{ADM}} = 10^4$ which was the optimal value achieving best overall performance.

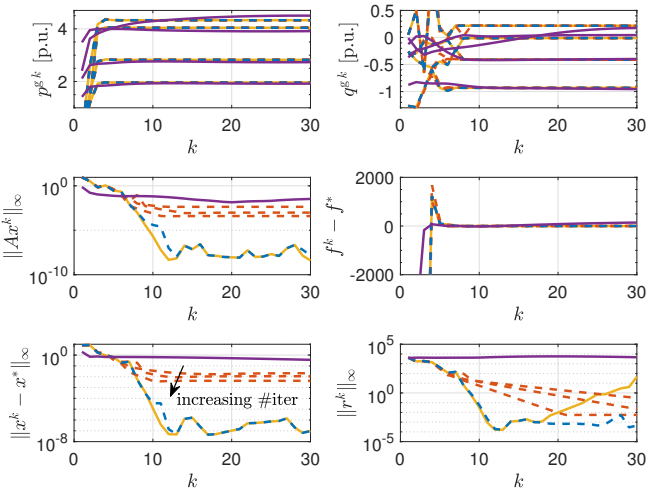


Figure 5.8: ALADIN (yellow), bi-level ALADIN with CG (dashed blue), bi-level ALADIN with ADMM (dashed orange) and ADMM (purple) for the 118-bus system.

Figure 5.8 shows the convergence behavior of basic ALADIN, b-ADMM, b-CG and pure ADMM. The graphs for ADMM and for basic ALADIN are the same as before but displayed for the sake of comparison. One can see that b-CG reaches almost the same overall performance compared with basic ALADIN. This is interesting as CG also provides a limited accuracy only in the inner step.

For b-ADMM on the other hand, this is different: the achievable accuracy of the whole algorithm seems to depend on the number of inner iterations in ADMM. This can be seen that $\|Ax^k\|_\infty$ and $\|x^k - x^\star\|_\infty$ stop decaying after around 10 outer iterations in Figure 5.8. Another difficulty when using ADMM as inner algorithm is tuning. We will investigate these two effects next.

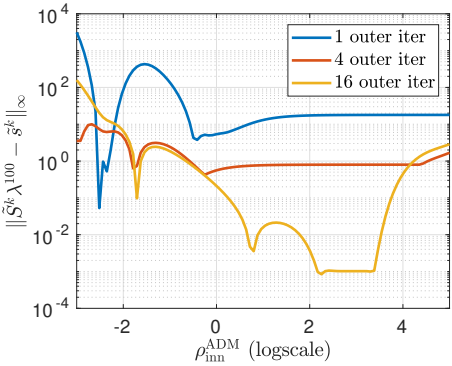


Figure 5.9: Accuracy of inner ADMM after 100 iterations for different values of $\rho_{\text{inn}}^{\text{ADM}}$ and after $k \in \{1, 4, 16\}$ outer iterations of bi-level ALADIN for 118-bus OPF.

5.5.2 Tuning inner ADMM

ADMM as an inner algorithm requires a serious amount of tuning. Figure 5.9 shows the accuracy in the inner problem (4.9), $\|\tilde{S}^k\lambda^n - \tilde{s}^k\|_\infty$, over the parameter $\rho_{\text{inn}}^{\text{ADM}}$ for a fixed amount of 100 inner d-ADMM iterations after $k \in \{1, 4, 16\}$ outer bi-level ALADIN iterations. One can see, that for the pair $(\tilde{S}^1, \tilde{s}^1)$, there are two different optimal values for $\rho_{\text{inn}}^{\text{ADM}}$ located at two different regions at around 10^{-2} and at around 10^0 after one outer iteration. This is one reason making tuning difficult.

Moreover, the optimal $\rho_{\text{inn}}^{\text{ADM}}$ changes significantly while ALADIN proceeds: after four outer ALADIN iterations, i.e. for $(\tilde{S}^4, \tilde{s}^4)$, the optimal value is located at about 10^0 and after 16 iterations it is located at around 10^2 to 10^3. This is a second reason making tuning of inner ADMM algorithm extremely difficult since this requires that ADMM has to be re-tuned after each outer ALADIN iteration.

A similar conclusion can be drawn from Figure 5.10. This figure shows the residual of the coordination step (4.9), $\|\tilde{S}^k\lambda^n - \tilde{s}^k\|_\infty$, for d-ADMM (Algorithm 6) and for d-CG (Algorithm 7) in the first ALADIN iteration (left) and after 16 ALADIN iterations (right). Also from this figure one can conclude that the optimal parameter for inner ADMM changes by four orders of magni-

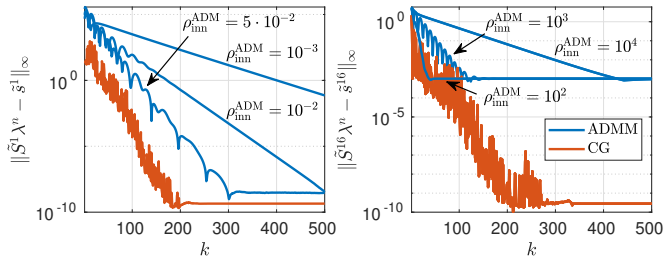


Figure 5.10: Convergence of d-CG and d-ADMM for the 118-bus OPF problem different values of $\rho_{\text{inn}}^{\text{ADM}}$ after one outer ALADIN iteration (left) and after 16 outer iterations (right).

tude from about 10^{-2} in the beginning to 10^2. Moreover one can see another interesting effect which is a possible explanation for the limited accuracy from Figure 5.10: whereas d-ADMM converges to a level of a bout 10^{-10} in the beginning of the ALADIN iterates, in later iterations the achievable accuracy seems to be limited to 10^{-3} Note that this can *not* be influenced by tuning of $\rho_{\text{inn}}^{\text{ADM}}$. We tried different tuning parameters and $\rho_{\text{inn}}^{\text{ADM}} = 10^2$ showed the best convergence and all other tuning parameters led to worse results. This effect seems interesting not to be investigated in the literature so far. In Appendix C we elaborate on this effect and we present an example, where this effect occurs already for very small problems.

For d-CG, in contrast, this effect does not occur and d-CG converges to high accuracies in all cases. Moreover, d-CG is parameter-free and does thus not require any tuning.

Remark 19 (Preconditioning for d-ADMM/d-CG) By preconditioning of (4.9), one can significantly accelerate convergence of d-ADMM and d-CG [GB17; Ste+20; Saa03]. However, note that preconditioning is also a centralized operation in general, where it has to be investigated whether the additional communication/coordination effort is outweighed by less iterations in d-ADMM/d-CG. In applications, where, the coefficient matrix $\tilde{S}^k$ does not change significantly from one step to another, offline preconditioning seems promising (e.g. in a control setting [Sta+16]). However, as we have shown previously, the coefficient matrix in bi-level ALADIN changes significantly and thus offline preconditioning seems less promising—in view of changes in the active set but

Table 5.4: Total forward communication until convergence (floats) for 118-bus OPF.

ALADIN variant	global $\delta = 10^{-2}$	local $\delta = 10^{-2}$	global $\delta = 10^{-4}$	local $\delta = 10^{-4}$
basic	852,992	0	1,066,240	0
condensed	13,152	0	16,440	0
bi-level (CG)	1,120	11,200	1,400	14,000
bi-level (ADMM-100)	0	24,000	-	-
bi-level (ADMM-200)	0	36,000	-	-
pure ADMM	0	2,840	0	6,320

also in view of the fact that curvature information changes in each ALADIN outer iteration.

5.6 Comparing communication

Next, we compare the forward communication from Section 4.6 for all algorithms numerically. Recall that in Section 4.6 we analyzed the communication effort *per iteration*. Here we analyze the *total amount of communication*, i.e. the number of floats which have to be communicated in sum until a pre-defined accuracy in the decision variables is reached. So here the "speed" of the algorithm plays a significant role since the amount of per-step communication is multiplied by the total number of iterations. We compare the algorithms for a given accuracy in the primal variables $\delta = \|x^k - x^\star\|_\infty$.

Table 5.4 and Table 5.5 show the total amount of forward communication for all investigated distributed optimization algorithms for the 118-bus OPF (Table 5.4) example and the 5-bus OPF example with line limits (Table 5.5). Hereby distinguish two different values for the accuracy δ for both grids: a low accuracy case and a high accuracy case. As predicted in Section 4.6, the condensed ALADIN variant significantly reduces the amount of required communication. For the 118-bus system, the reduction is quite large with a factor of round about 60 and for the 5-bus system with a factor of 5 a bit smaller. Note that the reduction here mainly depends on the number of coupling

Table 5.5: Total forward communication until convergence (floats) for 5-bus OPF with line limits.

ALADIN variant	global $\delta = 10^{-1}$	local $\delta = 10^{-1}$	global $\delta = 10^{-2}$	local $\delta = 10^{-2}$
basic	13,368	0	14,482	0
condensed	2,688	0	2,912	0
bi-level (CG)	1,820	10,920	2,240	13,440
bi-level (ADMM-100)	-	-	-	-
bi-level (ADMM-200)	-	-	-	-
pure ADMM	0	11,580	-	-

variables n_c in relation to the number of decision variables n_x. For the 5-bus system this ration is quite large making the reduction in communication smaller.

Moreover, one can observe that bi-level ALADIN with CG requires round about the same communication for the 118-bus system as the reduced space variant—but now this is mainly *local* communication. For the 5-bus system, b-CG requires slightly more communication than condensed ALADIN. b-ADMM needs an about a factor 2-3 larger amount of local communication mainly due to the slower convergence in the inner problem compared with b-CG.

Pure ADMM requires less communication compared with all other variants in the low-accuracy case ($\delta = 10^{-2}$) for the 118-bus system. However, as ADMM typically becomes quite slow in later iterations, the gap becomes smaller with higher accuracy requirements. For an accuracy of $\delta = 10^{-4}$ for example, the reduction factor of ADMM is only 2 instead of 5 for $\delta = 10^{-2}$.

For the 5-bus system with line limits (Table 5.5) this effect is extreme: as ADMM becomes very slow here (cf. Figure 5.5), ADMM needs more communication than b-CG. Moreover, the achievable accuracy of pure ADMM within $2,000$ iterations is no more than 10^{-2}. Moreover, b-ADMM did not converge at all for this case since the achiveable accuracy in the inner problem was not high enough to achieve an accuracy of 10^{-1}.

5.7 Summary and conclusion

The previous sections showed that ADMM can be applied successfully to OPF problems and converges quite robustly in many cases. However, ADMM has in general no convergence guarantee and divergence seems to occur in practice [Chr+17a]. Moreover, often convergence is possible only up to a limited accuracy and ADMM might require a large number of iterations to achieve even that accuracy. Although this level of accuracy is often sufficient in an OPF context as many parameters such as the power demands are anyways known only up to a limited certainty [Cap16], one has to be aware of the fact that a consensus violation $\|Ax\| \neq 0$ always means that the solution is infeasible to some degree. This implies that active/reactive power at interconnection points do not match perfectly and when applying ADMM to real systems and one has to account for that with and underlying controllers compensating this mismatch. Furthermore, we showed that the performance of ADMM can vary greatly depending on the type of constraints considered (with or without line limits for example).

We also showed that a feasible initialization in combination with high penalization parameters leads to feasible but not necessarily optimal solutions for ADMM. Moreover, we concluded that that consensus violation ($\|Ax^k\|_\infty$) alone, as sometimes used in the literature [GHT17; Ers15], is in general insufficient for termination. Especially in combination with the above combination of a feasible initialization with a feasible initial point this can lead to an arbitrary fast termination of ADMM. In this case, one can essentially enforce convergence of ADMM in one step in this case by choosing a sufficiently large ρ although the current iterate is far from a local minimizer. Moreover, the assumption of a feasible initial guess seems strong. Finding a feasible point has about the same complexity as the full OPF problem itself and again requires a strong central coordinator jeopardizing the goals of distributed optimization.

Applying ALADIN variants

As an alternative to ADMM, we proposed different ALADIN variants. ALADIN has the appealing property of a theoretically very fast local convergence, which we also observed in practice. One drawback of ALADIN are its comparably large communication and coordination requirements. As an alternative,

Table 5.6: ADMM, ALADIN and bi-level ALADIN for AC OPF.

	guarantees	speed	communication	coordination	robustness	accuracy
ADMM	−−	o	+	++	+	-
ALADIN	++	++	−−	−	o	++
bi-level ALADIN	++	++	+	+	o	++

we proposed a condensed variant of ALADIN, substantially reducing coordination and communication requirements. Moreover, with bi-level ALADIN, decentralized is also possible, while preserving ALADIN's fast local convergence properties. We showed that the performance of bi-level ALADIN with conjugate gradients (b-CG) is almost indistinguishable from basic ALADIN's performance although conjugate gradients introduce inexactness in the coordination step. For b-ADMM this is different: the achievable accuracy of b-ADMM seems to depend on the number of inner ADMM iterations. This comes from the fact that ADMM is able to solve the coordination problem up to a certain accuracy only and this effect can not be overcome by tuning. We also showed that this effect occurs already for very small-scale problems (cf. Appendix C). Moreover, tuning of inner ADMM can be difficult. However, b-ADMM offers full decentralization in the sense that it does not require any form of global scalar sum such as b-CG. The convergence rate of all the before mentioned ALADIN variants seems to be independent from the problem size and also from the considered constraints, which was not the case for ADMM. For large problems tuning of all ALADIN variants and also ADMM becomes increasingly difficult.

The properties of ADMM, ALADIN, and bi-level ALADIN are summarized in Table 5.6.

Choosing an algorithm

Choosing an appropriate algorithm is difficult. Whereas ADMM works well in many cases, convergence might be very slow and thus ADMM might take a long time to converge. The ALADIN variants usually converge very fast and decentralization is possible via bi-level ALADIN although tuning might become difficult for larger grids. If a high accuracy is required, ALADIN is

certainly first-choice. Moreover, ALADIN is guaranteed to converge for an initial point close enough to a local minimizer, which ADMM is not. In terms of total communication, ADMM has often a slightly lower footprint.

6 The ALADIN-α toolbox

In this chapter, we present an open-source MATLAB toolbox, ALADIN-α, implementing ALADIN, bi-level ALADIN and ADMM with a unified interface.

Major parts of this chapter are from [Eng+20]. We limit ourself to a brief overview here, for a more detailed description of options and subroutines we refer to [Eng+20] and to the documentation website [Eng20b].

6.1 Features of ALADIN-α

ALADIN-α is intended for rapid prototyping of distributed and decentralized optimization algorithms and aims at user-friendliness. The only user-provided information are objective and constraint functions—derivatives and numerical solvers are generated automatically by algorithmic differentiation routines and external state-of-the-art NLP solvers. A rich set of examples coming with ALADIN-α covers problems from robotics, power systems, sensor networks and chemical engineering underpinning its application potential.

ALADIN-α supports

- bi-level ALADIN and the condensed variant of ALADIN from Chapter 4;
- a BFGS Hessian extension from [Eng+19b];
- parametric NLPs enabling distributed Model Predictive Control (MPC);
- heuristics for regularization and parameter tuning.

The ADMM variant from Algorithm 4 is also included. ALADIN-α can be executed in parallel mode via the MATLAB parallel computing toolbox.

This can lead to a substantial speed-up, for example, in distributed estimation problems. However, although distributed optimization can be used for parallel computing, computation speed or real-time feasibility is currently *not* a primary focus of the toolbox. A documentation and many application examples of ALADIN-α are available under [Eng20b].

6.2 Literature review

Despite the various practical applications of distributed optimization, only very few software toolboxes are freely available. Even if one can find one of the rare examples, these tools are typically tailored to a specific application at hand and typically address *convex* problems. Examples are several versions of ADMM applied to a plethora of applications summarized in [Boy+11], with code available under [Boy+20]. An implementation of ADMM for consensus problems can be found under [Mot+20]. A tailored implementation of ADMM for OPF problems using an algorithm from [GHT17] can be found under [Guo20]. However, there is a lack of multi-purpose software tools for distributed optimization and we were not able to find any open-source implementations for generic distributed non-convex optimization problems. Also with respect to decentralized non-convex optimization, we were not able to find any publicly available code.

However, for parallel optimization efficient structure-exploiting tools exist. A closed-source parallel interior point software is `OOPS` [GG07]. The open-source package `qpDunes` is tailored towards parallel solutions of QPs arising in model predictive control [FSD15]. For general QPs, the partially parallelizable solver `OSQP` seems promising [Ste+20]. `PIPS` is a collection of algorithms solving structured linear programs, QPs, and general NLPs in parallel [CPZ14; Lub+11]. The software `HiOp` is tailored towards structured and very large-scale NLPs with few nonlinear constraints based on interior point methods [Pet19; PCA19]. Moreover, combining parallel linear algebra routines (e.g. `PARDISO` [Sch+01]) with standard nonlinear programming solvers (e.g. `IPOPT` [WB06]) also leads to partially parallel algorithms [Cur+12; KFS18]. All these tools are implemented in low-level languages such as `C` or `C++` leading to a high computational performance. However, their focus is mainly on computational

speedup via parallel computing rather than distributed and decentralized optimization in a multi-agent setting.

6.3 Parametric problem setup

ALADIN-α solves structured optimization problems of the form

$$\begin{aligned}
&\min_{x_1,\dots,x_{n_s}} && \sum_{i\in\mathcal{R}} f_i(x_i,p_i) && && && \text{(6.1a)}\\
&\text{subject to} && g_i(x_i,p_i)=0 && \mid \kappa_i, && \forall i\in\mathcal{R}, && \text{(6.1b)}\\
&&& h_i(x_i,p_i)\le 0 && \mid \gamma_i, && \forall i\in\mathcal{R}, && \text{(6.1c)}\\
&&& \underline{x}_i \le x_i \le \overline{x}_i && \mid \eta_i, && \forall i\in\mathcal{R}, && \text{(6.1d)}\\
&&& \sum_{i\in\mathcal{R}} A_i x_i = b && \mid \lambda, && && \text{(6.1e)}
\end{aligned}$$

which is a slight modification of (2.12) introducing box constraints separately. We do so for numerical efficiency reasons—some local solvers are able to handle box constraints by specialized routines. Moreover, problem (6.1) allows for parameter vectors $p_i \in \mathbb{R}^{n_{pi}}$. This can be useful for example in Model Predictive Control or if one would like to solve the same distributed problem multiple times for a variety of parameters.

6.4 Software structure

6.4.1 Code structure

In order to avoid side-effects and to make code-modification easy for beginners, we choose a procedual/functional programming style. We decided to implement all core features in MATLAB to enable easy rapid-prototyping. The overall structure of `run_ALADIN()`—which is the main function of ALADIN-α—is shown in Figure 6.1. First, a preprocessing step performs a consistency check of the input data and provides default options. The `createLocSolAndSens()` function initializes the local parameterized NLPs and sensitivities for all sub-

a) preprocessing	`checkInput(), setDefaultOpts()`
b) problem/sensitivity setup	`createLocSolAndSens()`
ALADIN main loop	`iterateAL()`
in parallel: c) solve local NLPs	`parallelStep(), BFGS(),`
in parallel: d) evaluate sensitivities	`parallelStepInnerLoop(),`
in parallel: e) Hessian approx./regularization	`updateParam(), regularizeH()`
f) solve the coordination QP	`createCoordQP(), solveQP(),solveQPdec()`
g) compute primal/dual step	`computeALstep()`
h) postprocessing	`displaySummary(), displayTimers()`

Figure 6.1: Structure of `run_ALADIN()` in ALADIN-α.

problems $i \in \mathcal{R}$. For constructing the local NLPs and sensitivities, we use `CasADi` due to its algorithmic differentiation features and the possibility to interface many state-of-the-art NLP solvers such as `IPOPT` [WB06; And+19]. `CasADi` itself relies on pre-compiled code making function and derivative evaluation fast. A `reuse` option allows reusing the `CasADi` problem setup. When the `reuse` mode is activated (e.g. when ALADIN-α is used within an MPC loop), `createLocSolAndSens()` is skipped resulting in a significant speed-up for larger problems.

In the ALADIN main loop `iterateAL()`, the function `parallelStep()` solves the local NLPs and evaluates the Hessian of the Lagrangian (or it's approximation e.g. when BFGS is used), the gradient of the objective, and the Jacobian of the active constraints (sensitivities) at the NLP's solution. Regularization is done in `parallelStep()` if needed. Moreover, in case the nullspace method or bi-level ALADIN is used, the computation of the nullspcaces and the Schur-complements is done locally shifting substantial computational burden from the centralized coordination step to `parallelStep()`. The function `updateParam()` computes dynamically changing ALADIN parameters for numerical stability and speedup.

The coordination QP is constructed in the function `createCoordQP()`. We use problem (2.25) including slack variables for numerical stability. Different dense and sparse solvers for solving the coordination QP are available in `solveQP()`. Most of them are based on solving the first-order necessary

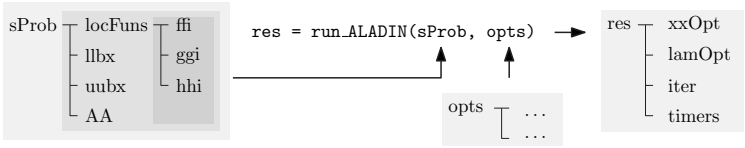


Figure 6.2: The `sProb` data structure for defining problems in form of (6.1).

conditions which is a linear system of equations. Available solvers are the MATLAB linear-algebra routines `linsolve(), pinv()`, the backslash operator and `MA57` based on the MATLAB LDL-decomposition routine. Using sparse solvers can speed up the computation time substantially. Note that only `MA57` and the backslash-operator support sparse matrices. The solver can be specified by setting the `solveQP` option. In case of convergence problems from remote starting points, it can help to reduce the primal-dual stepsize of the QP step by setting the `stepSize` in the options to a values smaller than 1.

6.4.2 Data structures

The main data structure for defining problems in form of (6.1) is a struct called `sProb` (cf. Figure 6.2). In this data structure, the objective functions $\{f_i\}_{i\in\mathcal{R}}$ and constraint functions $\{g_i\}_{i\in\mathcal{R}}$ and $\{h_i\}_{i\in\mathcal{R}}$ are collected in cells which are contained in a struct called `locFuns`. Furthermore, `sProb` collects lower/upper bounds (6.1d) in cells called `llbx` and `uubx`. The coupling matrices $\{A_i\}_{i\in\mathcal{R}}$ are collected in `AA`. Optionally, one can provide NLP solvers and sensitivities—in this case the problem setup in `createLocSolAndSens()` is skipped leading to a substantial speedup in runtime for larger problems. Optionally one specify initial guesses in `zz0` and initial Lagrange multipliers `lam0`. The second ingredient for ALADIN-α is an `opts` struct. There, one can specify the variant of ALADIN-α and algorithmic parameters. A full list of options with descriptions can be found under [Eng20b].

ALADIN-α returns a struct as output. This cell contains a cell of locally primal optimal solutions `xxOpt` with $\{x_i^\star\}_{i\in\mathcal{R}}$. `lamOpt` are the optimal lagrange multipliers for the consensus constraints (6.1e), $\lambda^\star$. Moreover the field `iter` contains information about the ALADIN iterates such as primal/dual iterates and `timers` contains timing information. Note that `run_ALADIN()`

and `run_ADMM()` have the same function signature in terms of `sProb`—only the options differ.

6.5 A tutorial example

Next, we provide two numerical examples illustrating how to use ALADIN-α in practice. The first one is a minimalistic example showing how to formulate problems in form of (6.1) and solving this problem with ALADIN-α.

First, we investigate how to reformulate a tutorial optimization problem in partially separable form 6.1. Let us consider the non-convex NLP

$$\min_{x_1,x_2\in\mathbb{R}} f(x) = 2\,(x_1-1)^2 + (x_2-2)^2 \tag{6.2a}$$

$$\text{subject to} \quad -1 \le x_1 \cdot x_2 \le 1.5. \tag{6.2b}$$

In order to apply ALADIN-α, problem (6.2) needs to be in form of (6.1). To get there, let us introduce auxiliary variables y_1, y_2 with $y_1 \in \mathbb{R}$ and $y_2 = (y_{21} \;\; y_{22})^\top$. Let us couple these variables again by introducing a consensus constraint $\sum_i A_i y_i = 0$ with $A_1 = 1$ and $A_2 = (-1 \;\; 0)$. Furthermore, let us reformulate the objective function f by local objective functions $f_1(y_1) := 2\,(y_1-1)^2$ and $f_2(y_2) = (y_{22}-2)^2$ with $f = f_1 + f_2$. Moreover, we reformulate the global inequality constraint (6.2b) by a local two dimensional constraint function $h_2 = (h_{21} \;\; h_{22})^\top$ with $h_{21}(y_2) = -1 - y_{21}\,y_{22}$ and $h_{22}(y_2) = -1.5 + y_{21}\,y_{22}$. Combining these reformulations yields

$$\min_{y_1\in\mathbb{R}, y_2\in\mathbb{R}^2} 2\,(y_1-1)^2 + (y_{22}-2)^2 \tag{6.3a}$$

$$\text{subject to} \quad -1 - y_{21}\,y_{22} \le 0, \quad -1.5 + y_{21}y_{22} \le 0, \tag{6.3b}$$

$$y_1 \;+\; (-1 \;\; 0)\, y_2 = 0, \tag{6.3c}$$

which is in form of problem (6.1). Note that the solutions to (6.2) and (6.3) coincide but (6.3) is of higher dimension, thus one can view the reformulation as a kind of lifting to a space of higher dimensionality. Moreover, observe that this reformulation contains a general strategy for reformulating problems in form of (6.1): if there is nonlinear coupling in the objective functions or the constraints, introducing auxiliary variables and enforcing them to coincide

```
% define symbolic variables
y1  =   sym('y1',[1,1],'real');
y2  =   sym('y2',[2,1],'real');

% define symbolic objectives
f1s =   2*(y1-1)^2;
f2s =   (y2(2)-2)^2;

% define symbolic ineq. constraints


h2s = [  -1-y2(1)*y2(2); ...
-1.5+y2(1)*y2(2)];

% convert symbolic variables to
% MATLAB functions
f1 = matlabFunction(f1s,'Vars',{y1});
f2 = matlabFunction(f2s,'Vars',{y2});

h1 = @(y1)[];
h2 = matlabFunction(h2s,'Vars',{y2});
```

```
% define symbolic variables
y_1 = SX.sym('y_1', 1);
y_2 = SX.sym('y_2', 2);

% define symbolic objectives
f1s = 2 * (y_1 - 1)^2;
f2s = (y_2(2) - 2)^2;

% define symbolic ineq.
   constraints
h1s = [];
h2s = [  -1 - y_2(1)*y_2(2); ...
-1.5 + y_2(1)*y_2(2)];

% convert symbolic variables to
% MATLAB functions
f1 = Function('f1', {y_1}, {f1s});
f2 = Function('f2', {y_2}, {f2s});

h1 = Function('h1', {y_1}, {h1s});
h2 = Function('h2', {y_2}, {h2s});
```

```
% define objectives
f1 = @(y1) 2 * (y1 - 1)^2;
f2 = @(y2) (y2(2) - 2)^2;

% define inequality constraints
h1 = @(y1) []
h2 = @(y2) [ -1 - y2(1) * y2(2);...
-1.5 + y2(1) * y2(2)];
```

Figure 6.3: Tutorial example with three different ways of problem setup.

by an additional consensus constraint in form of (6.1e) yields purely affine coupling. With that strategy, one can reformulate most nonlinear program in form of (6.1e).

Solution with ALADIN-α

```
% define coupling matrices
A1 = 1;
A2 = [-1, 0];

% collect problem data in sProb struct
sProb.locFuns.ffi = {f1, f2};
sProb.locFuns.hhi = {h1, h2};
sProb.AA          = {A1, A2};

% start solver with default opts
sol = run_ALADINnew(sProb);
```

```
================================================
==          This is ALADIN-alpha v0.1         ==
================================================
QP solver:        MA57
Local solver:     ipopt
Inner algorithm:  none

No termination criterion was specified.
Consensus violation: 6.6531e-12

Maximum number of iterations reached.

-------------   ALADIN-alpha timing   ----------
t[s]         %tot      %iter
Tot time......:     3.92
Prob setup....:     0.19         4.8
Iter time.....:     3.72         95
---------
NLP time......:     1.1                     29.7
QP time.......:     0.11                     2.8
Reg time......:     0.02                     0.6
Plot time.....:     2.27                    60.8

================================================
```

Figure 6.4: Collection of variables and output of ALADIN-α.

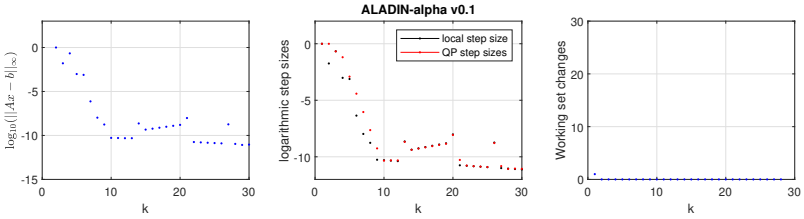


Figure 6.5: ALADIN-α iteration plot for tutorial problem (6.3).

To solve (6.3) with ALADIN-α, we set up our problem formulation in a struct `sProb` as described in Section 6.4.2. To illustrate different possibilities of problem setup for ALADIN-α, we construct the objective and constraints functions in three different ways: a), via the MATLAB symbolic toolbox, b), via the `CasADi` symbolic framework and, c), directly via function handles. All these ways are shown in Figure 6.3.

After defining objective and constraint functions, all function handles and the coupling matrices A_i are collected in the struct `sProb` (Figure 6.4). We call the `run_ALADIN()` function with an empty options struct leading to computation with default parameters. These steps and the resulting ALADIN-α report after running `run_ALADIN()` is shown on the right pane of Figure 6.4. In the ALADIN-α report, the reason for termination and timing of the main ALADIN-α steps is displayed. Note that plotting takes a substantial amount of time—so it advisable to deactivate online plotting if it is not needed for diagnostic reasons. Figure 6.5 shows the plotted figures while ALADIN-α is running. The figures show (in this order) the consensus violation $\|Ax - b\|_\infty$, the local step sizes $\|x^k - z^k\|_\infty$, the step size in the coordination step $\|\Delta x^k\|_\infty$ and the changes in the active set. From these figures one usually can recognize divergence quite fast and also can get a feeling on the effectiveness e.g. for new internal heuristics or the degree of accuracy reached after a certain number of iterations.

6.6 Applications beyond optimal power flow

ALADIN-α comes with a rich set of examples. These examples include

example	field	directory	online docs
		`examples/`	alexe15.github.io/ALADIN.m/
mobile robots	robotics/control	`robots`	robotEx
optimal power flow	power systems	`optimal_power_flow`	redComm
sensor network	estimation	`sensor_network`	ParallelExample
chemical reactors	chem./control	`chemical_reactor`	

Table 6.1: Application examples of ALADIN-α.

- an example for distributed optimal control of a chemical reactor, which we will present in the next section;
- an example for distributed optimal control of mobile robots [Meh+17; Eng+20];
- an example for distributed estimation in mobile sensor networks from Houska et al. [HFD16];
- a logistic regression example from machine learning.

Furthermore, we included several test problems form the Hock-Schittkowski test collection [HS80]. The code for all these examples is available in the `examples\` folder of ALADIN-α. Furthermore, we provide textual descriptions of these examples in the documentation of ALADIN-α online [Eng20b]. The application examples are summarized in Table 6.1.

6.7 Distributed control of a chemical reactor

Next, we show how to use ALADIN-α for distributed optimal control of a chemical reactor. This OCP can serve as a basis for distributed model predictive control [RMD17; SWR11; MA17]. The chemical process we consider here consists of two CSTRs and a flash separator as shown in Figure 6.6 [Cai+14; CLM11]. The goal is to steer the reactor to the optimal setpoint

$$u_s^\top = (0 \;\; 0 \; 0)$$

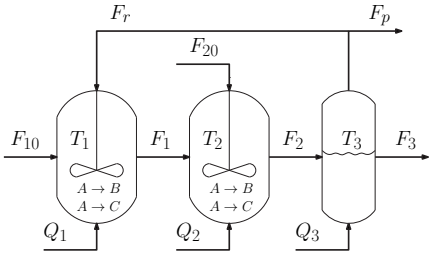


Figure 6.6: Reactor-separator process.

and

$$x_s^\top = (369.53 \;\; 3.31 \;\; 0.17 \;\; 0.04 \;\; 435.25 \;\; 2.75 \;\; 0.45 \;\; 0.11 \;\; 435.25 \;\; 2.88 \;\; 0.50 \;\; 0.12)$$

from an initial point

$$x(0)^\top = (360.69 \;\; 3.19 \;\; 0.15 \;\; 0.03 \;\; 430.91 \;\; 2.76 \;\; 0.34 \;\; 0.08 \;\; 430.42 \;\; 2.79 \;\; 0.38 \;\; 0.08).$$

After applying a fourth-order Runge-Kutta scheme for discretization, the dynamics of the CSTRs and the flash separator are given by

$$x_i^{k+1} = q_i(x_i^k, u_i^k, z_i^k) \quad \text{for} \quad i \in \mathcal{R},$$

where $q_i : \mathbb{R}^{n_{xi}} \times \mathbb{R}^{n_{ui}} \times \mathbb{R}^{n_{zi}} \to \mathbb{R}^{n_{xi}}$ are the dynamics of the ith vessel and where $\mathcal{R} := \{1, 2, 3\}$ is the set of vessels. Here, $x_i^\top = (x_{Ai}, x_{Bi}, x_{Ci}, T_i)$ are the states with x_{Ai}, x_{Bi} (10^3 mol/m^3) being the concentrations of the reactants, A, B and C and T (K) is the temperature. The inputs $u_i = Q_i$ (10^3 J/h) denote the heat-influxes of the individual vessel and z_i are copied states of the neighbored reactors influencing reactor i, i.e. $z_i := (x_j)_{j \in N(i)}$. Note that the feed-stream flow rates F_{10}, F_{20}, F_3, F_R and F_p are fixed and given. Detailed equations for the dynamics of the CSTRs/separator are given in [CLM11].

The optimal control problem

With the above, we are ready to formulate a discrete-time optimal control problem

$$\begin{aligned}
\min_{\substack{(x_i^k,\, z_i^k, u_i^k) \\ \forall k \in \mathbb{I}_{[1\ T]} \\ \forall i \in \mathcal{R}}} \; & \sum_{i \in \mathcal{R}} \sum_{k \in \mathbb{I}_{[1\ T]}} \frac{1}{2}(x_i^k - x_{is})^\top Q_i\,(x_i^k - x_{is}) + \frac{1}{2}(u_i^k - u_{is})^\top R_i\,(u_i^k - u_{is}) \\
\text{s.t.} \quad & x_i^{k+1} - q_i(x_i^k, u_i^k, z_i^k) = 0, \quad x_i^0 = x_i(0) \;\; \forall k \in \mathbb{I}_{[1\ T]},\ \forall i \in \mathcal{R}, \\
& \underline{u}_i \le u_i^k \le \bar{u}_i, \quad \underline{x}_i \le x_i^k, \quad \forall k \in \mathbb{I}_{[1\ T]},\ \forall i \in \mathcal{R}, \qquad (6.4) \\
& \sum_{i \in \mathcal{R}} A_i \left(x_i^{k\top}\ z_i^{k\top}\ u_i^{k\top}\right)^\top = 0 \quad \forall k \in \mathbb{I}_{[1\ T]},
\end{aligned}$$

with lower/upper bounds on the inputs $\overline{u} = -\underline{u} = (5 \cdot 10^4 \;\; 1.5 \cdot 10^5 \;\; 2 \cdot 10^5)^\top$ and lower bounds on the states $\underline{x}_i^k = 0$ for all times $k \in \mathbb{I}_{[1\ T]}$ and all vessels $i \in \mathcal{R}$. The weighting matrices are chosen to $Q_i = \mathrm{diag}(20 \;\; 10^3 \;\; 10^3 \;\; 10^3)$ and $R_i = 10^{-10}$. The matrices A_i are selected such that they represent the constraint $z_i := (x_j)_{j \in N(i)}$. The sampling time is $\Delta h = 0.01h$ and the horizon is $T = 10\,\mathrm{h}$. By defining $\tilde{x}_i^\top := \left(x_i^{k\top}\ z_i^{k\top}\ u_i^{k\top}\right)_{k \in \mathbb{I}_{[1\ T]}}$, $f_i(\tilde{x}_i) := \sum_{k \in \mathbb{I}_{[1\ T]}} \frac{1}{2}(x_i^k - x_{is})^\top Q_i\,(x_i^k - x_{is}) + \sum_{k \in \mathbb{I}_{[1\ T-1]}} \frac{1}{2}(u_i^k - u_{is})^\top R_i\,(u_i^k - u_{is}^k)$, $g_i(\tilde{x}_i) := \left(x_i^{k+1} - q_i(x_i^k, u_i^k, z_i^k)\right)_{k \in \mathbb{I}_{[1\ T-1]}}$, and $h_i(\tilde{x}_i) := \left((\underline{u}_i - u_i^k \;\; u_i^k - \bar{u}_i \;\; \underline{x}_i - x_i^k)^\top\right)_{k \in \mathbb{I}_{[1\ T]}}$ one can see that the OCP (6.4) is in form of (6.1) and thus solvable by ALADIN-α ($\tilde{x}_i$ here corresponds to x_i in (6.1)).

Numerical results

Figure 6.7 shows the resulting input and state trajectories for one OCP (6.4) for basic ALADIN and ADMM after 20 iterations, and for ADMM after 100 iterations. At first-glance all trajectories are quite close to each other. However, small differences in the input trajectories can be observed. Figure 6.9 shows the convergence indicators from Chapter 5 over the iteration index k. In logarithmic scale, these differences can be quite large. Fore example the

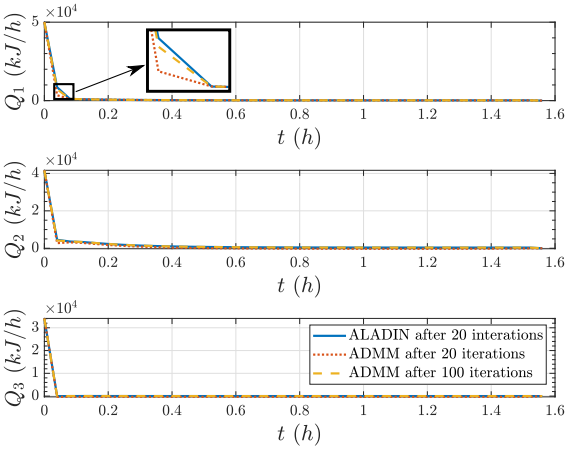


Figure 6.7: Optimal input trajectories computed by ALADIN & ADMM.

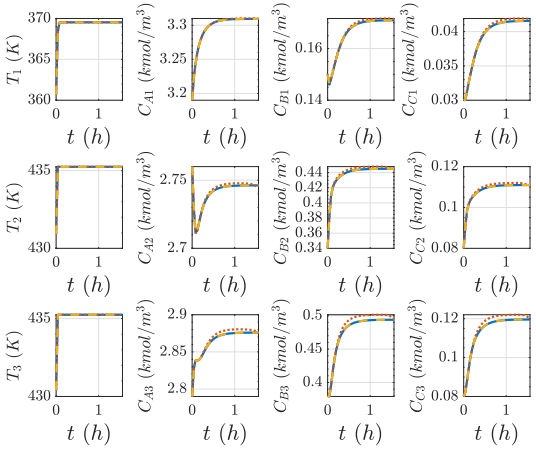


Figure 6.8: Optimal state trajectories computed by ALADIN & ADMM.

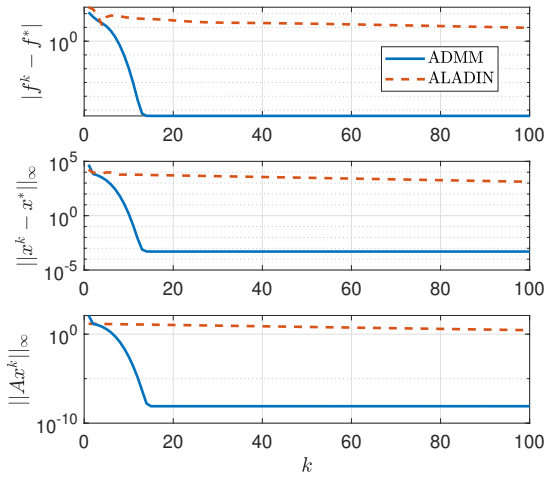


Figure 6.9: Convergence of ALADIN and ADMM for OCP (6.4).

consensus gap $\|Ax^k - b\|_\infty$ is in an order of 10^1 after 20 iterations which means that the physical values at the interconnection points have a maximum mismatch of 10^1. ALADIN converges quite fast and also to a very high accuracy. All trajectories were computed with `run_ALADIN()` and `run_ADMM()`.

7 Summary and Outlook

This thesis aimed at designing distributed and decentralized optimization algorithms for problems with non-convex constraints. In this context, a fast convergence under limited information exchange is key.

Chapter 2—Basics of Distributed Optimization

To build a foundation for algorithm development and for reviewing the literature of current distributed optimization methods, we briefly recalled distributed optimization algorithms such as the classical Alternating Direction Method of Multipliers (ADMM) and the more recent Augmented Lagrangian Alternating Direction Inexact Newton (ALADIN). We provided illustrative examples showing that classical algorithms such as ADMM might exhibit slow converge due to the lack of constraint information in the coordination step in combination with alternating projections. We also showed that ALADIN is able to overcome these limitations due to considering constraint information in its coordination step. Moreover, we emphasized that for certain non-convex problems, global convergence is out of reach—even for centralized algorithms.

Chapter 3—A survey on distributed optimization

We provided a literature review on distributed optimization in Chapter 3. Here, we showed that distributed and decentralized optimization algorithms from different communities are barely able to handle constrained non-convex problems from power systems and control we have in mind. To this end, we categorized the literature on distributed optimization along three main lines of research: primal algorithms, primal-dual algorithms and internal decomposition methods. Typically, the communities working on either of the above

research lines are mostly disconnected—this literature review attempted to take a more general perspective.

We showed that especially primal-dual algorithms and internal decomposition methods seem to be promising because of their effective constraint-handling capabilities. We classified ALADIN as a combination of primal-dual and internal decomposition methods combining the best of these two lines of research: distribution and fast convergence for non-convex problems. We emphasized that—despite of its very promising convergence properties—drawbacks of ALADIN are its high communication and coordination requirements, and a lack of decentralization.

Chapter 4—Bi-level Distributed ALADIN

In Chapter 4 we developed one of the first frameworks for decentralized optimizatio with convergence guarantees for problem with non-convex constraints. We presented two specific decentralized algorithms for distributing the coordination step of ALADIN: a decentralized variant of ADMM and a novel decentralized variant of the conjugate gradient algorithm. Decentralized conjugate gradient has the appealing property of a convergence in a finite number of steps, whereas state-of-the-art algorithms such as ADMM are often guaranteed to converge at a linear rate at most and the convergence modulus can be slow. Decentralized conjugate gradients and decentralized ADMM introduce inexactness into the coordination step of ALADIN. We showed mathematically that, despite this inexactness, the very fast local convergence properties of ALADIN can be preserved if the error in the coordination step decays fast enough.

Possible directions for future work

As bi-level ALADIN is guaranteed to convergence locally at present, further research on distributed and decentralized globalization routines seems to be important. Moreover, the amount of tuning one has to invest typically increases with the problem size. Globalization routines and internal auto-tuning routines are thus promising candidates to address these challenges.

The amount of communication and the overall convergence of bi-level ALADIN depends on the accuracy and thus on the number of inner iterations in the coordination step. Distributed preconditioning might help here to reduce the number of inner iterations. Similarly, a dynamic adjustment of the termination criterion in the coordination step might help to decrease the total communication requirements of by using less inner iterations.

Chapter 5—Application to Power Systems

In Chapter 5, we evaluated the performance of bi-level ALADIN on optimal power flow problems up to several hundred buses. We started by reviewing relevant literature in this context and concluded that current approaches either have no convergence guarantee for AC OPF, use oversimplified models or relaxations, where the solution to the original problem might not be recoverable from the relaxed problem. Other algorithms require a large amount of central coordination.

We proposed ALADIN and bi-level ALADIN as alternatives and compared their performance ADMM as one of the most-frequently used distributed optimization methods for OPF. We showed that bi-level ALADIN is able to converge much faster than ADMM—also for a limited amount of inner iterations. Moreover, we highlighted that bi-level ALADIN with conjugate gradients is able to reach about the same low communication footprint of ADMM for certain problems, while converging to a much higher accuracy.

We also showed that for bi-level ALADIN with ADMM as an inner algorithm, the achievable accuracy is often limited due to the limited accuracy achievable by ADMM in the inner loop. This effect seems not to be investigated in the literature so far. Moreover, we showed that tuning of inner ADMM might be difficult since the optimal penalization parameter changes with the outer ALADIN iterations.

In the distributed OPF literature, high penalization parameters in combination with a feasible initial point is sometimes used for ADMM. We showed numerically and mathematically that, in its extreme, this combination makes ADMM getting stuck at the initial point. We also showed that pure ADMM is able to converge only up to a limited accuracy for the problems we consider.

Possible directions for future work

The considered OPF problem forms the basis for many other problems such as reactive power dispatch problems, state estimation problems, power flow problems, and redispatch problems [Du+19; Mur+18] all of which might benefit from distributed optimization. Also multi-stage OPF problems, where OPF problems are coupled in time via storages might be of interest [FE19]. Broadening the view from power systems to energy systems, application in other domains such as building control [Su+20; Zwi+19] seems promising as also here a limited information exchange and decentralized computation are often desirable.

Investigating the slow convergence of ADMM for certain problem classes and particularly in context of OPF in more detail seems to be important. Especially the modulus switching from Appendix C is key, since slow convergence can be observed even for very small problems without constraints. This is relevant for bi-level ALADIN, as ADMM as inner algorithm suffers from this slow convergence and make the achievable accuracy of bi-level ALADIN with ADMM limited for OPF.

Tuning of bi-level ALADIN becomes increasingly difficult with a growing problem size—also for problems from power systems. Hence, testing new globalization routines and internal heuristics on problems from power systems seems worth investigating.

Chapter 6—The ALADIN-α toolbox

In Chapter 6, we presented one of the first general-purpose open source toolboxes for decentralized non-convex optimization named ALADIN-α implementing the algorithms from the preceding chapters. The toolbox enables rapid-prototyping of distributed and decentralized algorithms in a modular framework. The MATLAB toolbox comes with a rich set of code examples for distributed and decentralized optimization from different engineering fields ranging from power systems, via chemical engineering, to mobile sensor networks and robotics to machine learning, highlighting its broad applicability.

We investigated numerical performance of ALADIN in comparison with ADMM on an optimal control problem for a three-vessel chemical reactor.

Moreover, ALADIN-α comes with advanced features such as parallel computing and parametric programming enabling its usage in context of distributed model predictive control.

Possible directions for future work

It seems promising to develop a generalized variant of ALADIN-α, where the user can pass own solvers and differentiation routines eliminating the dependency on the automatic differentiation tool CasADi. This might help to speed up computation—especially for large problems.

For optimal control, a real-time implementation of ALADIN-α with code generation seems promising. An implementation in free languages such as Julia or Python also seems important to enlarge the possible user base.

Closing remarks

With bi-level ALADIN, we presented one of the first families of algorithms for decentralized optimization with non-convex constraints. We showed that bi-level ALADIN is guaranteed to converge fast under a limited information exchange. We demonstrated that these properties also hold in practice for relevant problems from power systems and control. Moreover, we presented one of the first toolboxes for decentralized non-convex optimization implementing the algorithms from this thesis.

However, these are merely first steps and future will tell which algorithms work robustly for a broad variety of problems.

A Mathematical Background

A.1 Mathematical Basics

We recall some basic mathematical results, which we used in our derivations.

Matrix norms

Recall that the following properties of matrix/vector norms for matrices $A, B \in \mathbb{R}^{n\times m}$ and scalars $\alpha \in \mathbb{R}$ must hold

$$\|A\| \geq 0 \qquad \text{(positive-valuedness)}, \tag{A.1a}$$
$$\|\alpha A\| = |\alpha|\|A\| \qquad \text{(absolute homogenity)}, \tag{A.1b}$$
$$\|A + B\| \leq \|A\| + \|B\| \qquad \text{(triangular inequality)}, \tag{A.1c}$$
$$\|A\| = 0 \text{ iff } A = 0 \qquad \text{(definiteness)}. \tag{A.1d}$$

Note that we assume in all our derivations that matrix norms are *induced* by their corresponding vector norm, i.e. for a given vector norm $\|\cdot\|$ on $\mathbb{R}^n$, the induced matrix norm for a matrix $A \in \mathbb{R}^{n\times m}$ with respect to $\|\cdot\|$ is $\|A\| := \sup_{x\in\mathbb{R}^n, \|x\|=1} \|Ax\|$. For induced matrix norms, the following property holds additionally

$$\|AB\| \leq \|A\|\|B\| \qquad \text{(sub-multiplicativity)}.$$

Moreover we will need that given two matrices $A \in \mathbb{R}^{n\times m}, 0 \in \mathbb{R}^{n\times l}$, we have $\|(A \quad 0)\| = \sup_{\|(x^\top \; y^\top)\|=1} \|(A \quad 0)(x^\top \quad y^\top)^\top\| = \sup_{\|x\|=1} \|Ax\| = \|A\|$, where the last step follows from the fact that the fact that the choice of y does not change the supremum of $\|Ax\|$.

Another important property which we will use is the "triangular inequality for integrals" for integrable functions $F : \mathbb{R} \to \mathbb{R}^n$,

$$\left\| \int_a^b F(t)dt \right\| \leq \int_a^b \|F(t)\|dt, \tag{A.2}$$

which can be shown by the above properties and the definition of the Riemann integral.

Definiteness of matrices

A matrix $M \in \mathbb{R}^{n\times n}$ is called *positive definite* if $x^\top Mx \geq 0$ for all $x \in \mathbb{R}^n$. If the before inequality holds strictly, M is called *strictly positive definite*. Equivalently, we write $M \succeq 0$ and $M \succ 0$.

Fundamentals from calculus

By the *fundamental theorem of calculus* we have for any continuously-differentiable function $g : [a\ b] \to \mathbb{R}$

$$\int_{t_1}^{t_2} g'(t)dt = g(t_1) - g(t_2).$$

For an extension to the multivariate case let us consider a continuously differentiable vector-valued function $f : \mathbb{R}^n \to \mathbb{R}$. Define $g(t) := f(a + t(b - a))$ and thus $g'(t) = \nabla f(a + t(b - a))^\top (b - a)$. Then we have

$$\int_0^1 \nabla f(a + t(b - a))^\top (b - a)\, dt = f(b) - f(a).$$

The extension to a vector field $F : \mathbb{R}^n \to \mathbb{R}^m$ is done by concatenation of single-valued functions $f_i : \mathbb{R}^n \to \mathbb{R}$ as $F(x) := (f_1(x), \ldots, f_m(x))^\top$ yielding

$$\int_0^1 \nabla F(a + t(b - a))^\top (b - a)\, dt = F(b) - F(a), \tag{A.3}$$

for some $a, b \in \mathbb{R}^n$, where ∇F is the Jacobian of F and the integral is evaluated component-wise.

Convergence of sequences

In context of optimization algorithm, two question are of significant interest: first of all, the question "Does the sequence generated by the algorithm converge to a minimizer/stationary point?" and, secondly "If it converges, how fast does it converge to that point?".

Let us address the first question. A sequence $\{x^k\}$ in the metric space $(\mathbb{R}^n, d(x, y) = \|x - y\|)$ is called *convergent* to a point $x^\star$ if for any $\epsilon > 0$ one can find an N such that $\|x^k - x^\star\| < \epsilon$ for all $n > N$. Not that for a sequence satisfying $\|x^{k+1} - x^\star\| \leq c\|x^k - x^\star\|$ with $c \in [0\ 1)$ one can always find such an N, since by iterating this equality we have $\|x^N - x^\star\| \leq c^N\|x^0 - x^\star\| \leq \epsilon$ by choosing an N large enough. Thus, one can show convergence to $x^\star$ by showing that the above inequality holds.

To characterize the "speed" of convergence we distinguish between four basic convergence rates (with increasing speed): Q-sublinear, Q-linear, Q-superlinear and Q-quadratic convergence. The "Q" here stands for quotient convergence. Consider a sequence $\{c_k\}_k$ that converges to c. We say that this sequence is *Q-linearly convergent* to c with modulus $\mu \in (0,\ 1)$ if[1]

$$\limsup_{k\to\infty} \frac{|c_{k+1} - c|}{|c_k - c|} = \mu.$$

We say that $\{c_k\}$ is *Q-sublinearly convergent* if the above limit converges to 1 and *superlinearly convergent* if it converges to 0. We say that $\{c_k\}$ converges *Q-quadratically* to c if

$$\limsup_{k\to\infty} \frac{|c_{k+1} - c|}{|c_k - c|^2} = \mu < \infty.$$

[1] Linear convergence essentially means exponential convergence in the sense that a linearly convergent series can be upper bounded by a exponentially covergent sequence. The name "linearly" convergence comes from the fact that a linearly convergent series looks linear in a semilogarithmic plot.

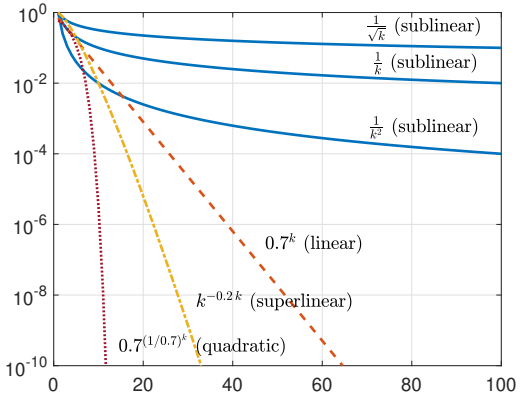


Figure A.1: Convergence rates in comparison.

Note that in view of the above definition of linear convergence, if we can find a sequence satisfying $|c^{k+1} - c| \leq \mu |c^k - c|$, with $\mu \in (0, 1)$, linear convergence with modulus μ immediately follows. Similarly, we can conclude quadratic convergence from an inequality $|c^{k+1} - c| \leq \mu |c^k - c|^2$. Figure A.1 shows examples of sublinear, linear, superlinear and quadratically convergent sequences.

Note that there is an issue in the above definitions in case the sequence can only be upper bounded by a fast convergent sequence but does not decrease in *each* step in the above sense. This might be problematic since some optimization methods do not produce a descent in the objective function value in each step. Consider for example the sequence $c^k = e^{-k}(1 + \sin(10\,k\,\frac{\pi}{2}))$. This sequence somehow converges "intuitively linearly" to zero but not in the sense of the above definition since $\limsup_{k\to\infty} \frac{e^{-(k+1)}(1+\sin(10\,(k+1)\,\frac{\pi}{2}))}{e^{-k}(1+\sin(10\,k\,\frac{\pi}{2}))} = \frac{e^{-1}(1+\sin(10\,(k+1)\,\frac{\pi}{2}))}{(1+\sin(10\,k\,\frac{\pi}{2}))} > 1$.

Therefore, it makes sense to use a weaker notion of convergence rate, the *root* convergence (R-convergence), where convergence is characterized by a majorizing, upper bounded, and Q-convergent sequence $\{\bar{c}_k\}$. In view of that we say that the sequence $\{c_k\}$ *converges R-{linearly, superlinearly,*

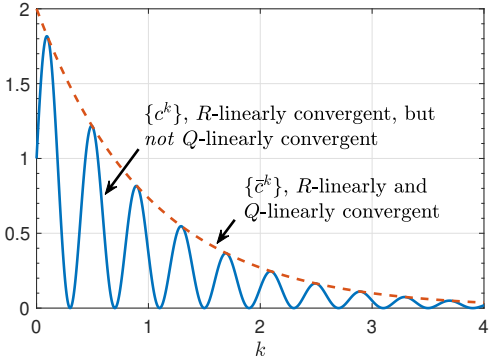


Figure A.2: R-convergence vs. Q-convergence.

quadratically} to c if there exists a Q-{linearly, superlinearly, quadratically} sequence $\{\bar{c}_k\}$ converging to zero such that

$$|c^k - c| \leq \bar{c}^k.$$

For the above example we can the define the Q-linearly convergent sequence $\bar{c}^k = 2e^{-k} \geq e^{-k}(1 + \sin(10\,k\frac{\pi}{2}))$ and thus R-linear convergence of $\{c_k\}$ follows. Figure A.2 illustrates the above example. Note that trivially each Q-linearly convergent sequence is R-linearly convergence but the converse is not true.

R-convergence can be indicated using the order notation. For the above example we can simply write $|c^k - c| = O(e^{-k})$ indicating R-linear convergence of $\{c_k\}$ to c. We say that a function (sequence) $f : \mathbb{R} \to \mathbb{R}$ is of order $g : \mathbb{R} \to \mathbb{R}$ with respect to $a \in \mathbb{R} \cup \{-\infty, \infty\}$ if there exists a $c > 0$ such that $|f(x)| \leq c|g(x)|$ for $x \to a$. Here we usually mean the limiting behavior with repsect to $a = \infty$ but in optimization this notion is also sometimes used for describing residuals in Taylor series expansions where a is the point of expansion.

Moreover, there exist several ways of characterizing the convergence of optimization methods. They can be characterized in terms of a distance to a local minimizer (or a stationary point), i.e. $\|x^k - x^\star\| = O(\cdot)$, in terms of the

objective function value $|f(x^k) - f(x^\star)| = O(\cdot)$ or in terms of stationarity of the objective function (in the unconstrained case) $\|\nabla f(x^k)\| = O(\cdot)$. In the present work, we refer to the first case, i.e. convergence estimates considering the distance to a local minimizer if not stated differently. Note that, however, particularly in context of distributed optimization, convergence is often defined in terms of the objective function value. Under certain conditions such as Lipschitz continuity or strong convexity one can transfer estimates in terms of the objective function to a result in terms of the distance to a local minimizer [Ber99, Chap 1.2] [Bec17] but not always. Sometimes convergence results are stated in terms of a specific accuracy, i.e. the number of iterations needed to reach a specific accuracy in the objective function value $|f(x^k) - f(x^\star)| < \epsilon$ [Bec+18] which we will not use in the present work.

A.2 Distributed problem formulations

Here, we give a brief overview on common problem formulations used in the literature. Recall that in the present work we consider problems in form of (2.12),

$$\min_{x_i,\dots,x_R} \sum_{i\in\mathcal{R}} f_i(x_i) \tag{A.4a}$$

$$\text{subject to} \quad g_i(x_i) = 0, \qquad \forall i \in \mathcal{R}, \tag{A.4b}$$

$$h_i(x_i) \leq 0, \qquad \forall i \in \mathcal{R}, \tag{A.4c}$$

$$\sum_{i\in\mathcal{R}} A_i x_i = b, \tag{A.4d}$$

where f_i,g_i and h_i are smooth. A common technique to simplify problem (A.4) is to replace f by $\tilde{f}_i := f_i + \iota_{\mathcal{X}_i}$, where ι is the indicator function of the set $\mathcal{X}_i := \{x_i \in \mathbb{R}^{n_{xi}} \mid g_i(x_i) = 0,\ h_i(x_i) \leq 0\}$, cf. Section 2.2.1. This yields an equivalent problem

$$\min_{x_1,\dots,x_R} \sum_{i=1}^{N} \tilde{f}_i(x_i) \tag{A.5a}$$

$$\text{subject to} \quad \sum_{i\in\mathcal{R}} A_i x_i = b. \tag{A.5b}$$

Note that if all g_i are affine and all h_i are convex, all $\tilde{f}_i$s are convex but possibly non-differentiable and discontinuous functions, which render derivative-based algorithms such as gradient-based methods or SQP methods non-applicable. However, some algorithms such as ADMM can handle such discontinuities under certain circumstances. By introducing auxiliary variables $z_i \in \mathbb{R}^{n_{xi}}$, we can write (A.5) as

$$\begin{aligned} &\min_{x,z} \ \tilde{f}(x) + \iota_C(z) \\ &\text{subject to } x_i - z_i = 0, \qquad \forall i \in \mathcal{R} \end{aligned}$$

with $\tilde{f} = \sum_{i\in\mathcal{R}} \tilde{f}_i$ and where $C = \{z \in \mathbb{R}^{n_x} \parallel \sum_{i\in\mathcal{R}} A_i z_i - b = 0\}$. This problem is now in the famous *two-block* form

$$\min_{x,z} \ f(x) + g(z) \tag{A.6}$$

$$\text{subject to } Ax + Bz = c, \tag{A.7}$$

used in many contexts of distributed optimization [BT89, Chap 3.4], [Bec17, Chap 10], [PB14; EB92; Och+14; BT09; DLS16; GB16; Boy+11]. This form is often used in so-called *operator splitting* schemes having a different view on distributed optimization from a operator-theoretic perspective [EB92; GOY17; BC11]. Note that the ADMM is shown to be a special case of the so-called *Douglas-Rachford*-splitting algorithm [Gab83]. Generally, splitting schemes can often efficiently exploit the fact that one of the two functions f or g is separable which is also here case for $\tilde{f}$. Note that these splitting schemes often rely on convexity which can be ensured by choosing convex f_i and h_i and affine g_i.

Consensus problems

In many cases, distributed optimization approaches consider unconstrained problems in form of

$$\min_{x} \sum_{i\in\mathcal{R}} f_i(x). \tag{A.8}$$

in the literature [Shi+14; Shi+15b; NO09; MO17; DLS16; NOP10; Boy+11; ZK14; NOS17]. Here, the f_i are typically continuous but they might not be differentiable, i.e. they might consider non-smooth components but do typically not encode constraints via the indicator function as in the previous subsection. Note that here all f_i depend on *one common* decision vector x. Similar to the above, by introducing R copies of x, $z_i \in \mathbb{R}^{n_x}$, it is possible to transform (A.8) into so-called *consensus form*

$$\min_{x,z_i} \sum_{i\in\mathcal{R}} f_i(z_i) \tag{A.9a}$$

$$\text{subject to } z_i = x \quad \text{for all} \quad i \in \mathcal{R} \tag{A.9b}$$

which is again in two-block form (A.6). Note that in this formulation, the network structure is not explicitly considered which makes this form interesting mostly in parallel computing contexts. Next, we will review problem structures considering the network structure.

Connection to optimization over networks

Many works consider distributed optimization over networks [DLS16; Shi+14; Shi+15b; NO09; TT17]. Therein, problem (A.8) is typically reformulated as consensus problems but subject to certain communication constraints which are encoded as a graph $G = \{\mathcal{N}, \mathcal{E}\}$. One option for doing so [Shi+14] is to introduce global auxiliary variables z_{ij} only for the edges $(i, j) \in \mathcal{E}$ yielding

$$\min_{x_i,z_{ij}} \sum_{i\in\mathcal{R}} f_i(x_i) \tag{A.10}$$

$$\text{subject to } x_i = z_{ij}, \quad x_j = z_{ij} \text{ for all } (i, j) \in \mathcal{E}. \tag{A.11}$$

Note that (A.10) is again in two-block form (A.6). An alternative option is copying all variables and enforcing consensus directly via characteristic

matrices of G such as the Laplacian matrix $L \in \mathbb{R}^{|\mathcal{N}|\times|\mathcal{N}|}$ or its incidence matrix $I \in \mathbb{R}^{|\mathcal{N}|\times|\mathcal{E}|}$ [Dor18] for one-dimensional problems. This yields

$$\min_{x_i} \sum_{i\in\mathcal{R}} f_i(x_i) \tag{A.12}$$

$$\text{subject to } Lx = 0 \quad \text{or} \quad I^\top x = 0. \tag{A.13}$$

Applying ADMM or a gradient/subgradient method reveals that it suffices to exachange information only between neighbors, i.e. over the edges $(i, j) \in \mathcal{E}$. Note that due to these reformulations, properties of G such as connectedness or algebraic connectivity influence the convergence properties of the resulting decentralized algorithms, where stronger connectivity typically leads to faster convergence [Shi+14].

Algorithms for (A.8) are developed without explicitly introducing the graph properties into the problem formulation [NO09; Shi+15b; NOS17; DLS16; YLY16]. In these works, typically matrices W encoding connectivity information are directly introduced into the iteration schemes e.g. of a gradient method. Recent overviews on optimization over networks can be found in [NOW18; NOR18], cf. [Dor18] for an introduction from a control-theoretic perspective.

The here-used connectivity encoding technique

In the present work we use network encoding which is closets related to the edge-based technique form the previous paragraph. Our formulation is more general as the before-introduced techniques in the sense that it allows for multiple decision variables per node, where all these decision vectors can be of a different dimension. This already shows that in the context of our work we mainly rely on "complexity in the nodes" in the sense that usually we have few nodes, each of which has a quite complex subproblem. Rather than defining the graph a-priori, we follow a different route: we define our communication graph based on the sparsity in the consensus matrices $\{A_i\}$. In view of (A.4), we have a node set $\mathcal{R}$ to each of which we assign a decision vector x_i, for all $i \in \mathcal{R}$. We define the edges of the graph via the non-zero entries in $\{A_i\}$: two subsystems $i, j \in \mathcal{R}$ are called *neighbored* (and thus we assign an edge (i, j) to the set $\mathcal{E}$) if there is a row $c \in \mathcal{C}$ which is non-zero in bot corresponding

A_i and A_j. This way, our formulation is more general compared to the other formulations above since we allow for multiple variables per node and for multiple interconnections between two nodes.

Moreover, other couplings exists such as objective coupling, where the coupling between variables takes place in the objective function. However, typically they can be reformulated in constrained-coupled form. We refer to [NND11] for a more detailled treatment in context of control.

A.3 Consensus reformulation preserves LICQ

In order to use distributed optimization methods, it is often necessary to reformulate problems in affinely-coupled separable form (2.12) as we have seen for OPF in Section B.4. One approach for doing so is copying decision variables which couple the subsystems nonlinearly and coupling them again by an additional affine equality constraints which then immideately leads to problems in form of problem (2.12). In this case, an important question is whether or not we "destroy" certain nice properties of the original problems such as constraint qualifications from Definition 2. Next, we show that this is not the case, i.e. that copying variables and adding additional affine coupling constraints preserves LICQ.

Let us assume we have an optimization problem with two blocks of variables $x \in \mathbb{R}^{n_x}, y \in \mathbb{R}^{n_y}$ subject to equality constraints $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \to \mathbb{R}^{n_g}$ with $n_g \leq n_x + n_y$. We now want to show that a introducing auxiliary variables $z \in \mathbb{R}^{n_y}$ preserves LICQ.

If LICQ holds for g, we have

$$\operatorname{rank}\left(\nabla g(x, y)\right) = n_g$$

in a small neighborhood of $(x^\star, y^\star)$. If we minimize over an extended set of variables $(x, y, z) \in \mathbb{R}^{n_x+2n_y}$, the constraints are given by

$$\tilde{g}(x, y, z) := \begin{pmatrix} g(x, z) \\ y - z \end{pmatrix} = 0.$$

Moreover

$$\nabla \tilde{g}(x,y,z) = \begin{pmatrix} \nabla_x \tilde{g}(x,y,z) & 0 & \nabla_z \tilde{g}(x,y,z) \\ 0 & I & -I \end{pmatrix}. \tag{A.14}$$

By LICQ, the first row of (A.14) has rank n_g. Using the identity

$$\operatorname{rank}\,(A^\top \; B^\top)^\top = \operatorname{rank}(A) + \operatorname{rank}(B - BA^\top A)$$

then yields $\operatorname{rank}\,(\nabla g(x,y,z)) = n_g + n_y$. This shows that LICQ is preserved.

A.4 Proof of Theorem 4

The proof is similar to standard proofs for Newton-type methods from [Die16; NW06; Ber99] and given here for the sake of a self-contained presentation and because of it's importance for the convergence analysis of ALADIN. The idea

is to estimate the distance to a local primal-dual solution to (2.12) $q^\star$. With the Newton-type iteration (2.7) we get

$$\begin{aligned}
\|q^{k+1} - q^\star\| &= \|q^k - q^\star - (M^k)^{-1} F(q^k)\| \\
&= \|q^k - q^\star - (M^k)^{-1}(F(q^k) - F(q^\star))\| \\
&= \|(M^k)^{-1} M^k (q^k - q^\star) \\
&\qquad - (M^k)^{-1} \int_0^1 \nabla F\big(q^\star + t(q^k - q^\star)\big)(q^k - q^\star) dt\| \\
&= \|(M^k)^{-1}(M^k - \nabla F(q^k))(q^k - q^\star) \\
&\quad - \int_0^1 (M^k)^{-1} \Big(\nabla F\big(q^\star + t(q^k - q^\star)\big) - \nabla F(q^k)\Big)(q^k - q^\star) dt\| \\
&\leq \left\|(M^k)^{-1}(M^k - \nabla F(q^k))\right\| \left\|q^k - q^\star\right\| \\
&\quad + \int_0^1 \left\|(M^k)^{-1} \Big(\nabla F\big(q^\star + t(q^k - q^\star)\big) - \nabla F(q^k)\Big)\right\| dt \left\|q^k - q^\star\right\| \\
&\leq \kappa \left\|q^k - q^\star\right\| + \int_0^1 \omega \|q^k - q\| dt \left\|q^k - q^\star\right\| \\
&= \left(\kappa + \frac{\omega}{2}\|q^k - q^\star\|\right) \|q^k - q^\star\|,
\end{aligned}$$

where we used (in this order) $F(q^\star) = 0$, the fundamental theorem of calculus (A.3), continuous differentiability of F and adding $\pm(M^k)^{-1}\nabla F(q^k)(q^k - q^\star)$ and pulling it into the integral, (in one step) the triangular inequality—(A.2)—submultiplicativity—and standard properties of integrals, the Lipschitz condition (2.10a) and compatibility condition (2.10b) and the positivity of norms, and finally evaluation of the integral. Convergence follows by inserting $\|q^k - q^\star\| < \frac{2(1-\kappa)}{\omega}$ into the contraction estimate leading to $\|q^{k+1} - q^\star\| < \|q^k - q^\star\|$ which is sufficient for convergence to $q^\star$. □

B Power System Fundamentals

Using a sufficiently accurate grid models is extremely important. In this section, we will provide a brief introduction to the *AC power system model* providing a sufficient accuracy for decision making on a seconds-to-hours time scale. This model describes the nonlinear relation between the injected/consumed powers at all nodes and the corresponding voltages. One often-employed alternative is the so-called *DC-model* which is often employed in context of power system optimization.[1] Note that this model is often insufficient due to the lack of voltage and reactive power modeling.

B.1 The AC model

Let us make the assumption we use for the AC model explicit. We assume

(i) time-scale separation;

(ii) sufficiently short transmission lines;

(iii) the absence of nodal shunts and transformers;

(iv) a symmetric power system;

(v) and a constant-power load model.

[1] The DC models assumes constant voltages, no resistances in transmission lines and small voltage angles yielding a *linear model* of power systems enabling to use all benefits from convex optimization. However, while these assumptions represent the physical laws in transmission grids quite well, in distribution grids they are often violated to a large degree for example due to high resistances and high power injections from renewables at weak nodes leading to quite large voltage angle differences [WW13].

Assumption (i) implies that we have a steady-state and transient phenomena can be neglected; assumption (i) together with assumption (ii) allows for using a π-equivalent model of transmission-lines/cables. Assumption (iii) is mainly made for simplified presentation, including shunts and transformers poses no significant difficulty from an algorithmic perspective—however it somewhat complicates model derivation. Assumption (iv) implies that we can represent all three phase by their single-phase equivalent. Assumption (v) is made to be able to avoid the difficulty of handling different load models which is quite common under for the here-considered steady-state problems [Ari+18].

Remark 20 (Discussion of the above assumptions) Finer-grained models exists allowing to violate the above assumptions. Examples include using hyperbolic equations for the transmission lines (or even partial differential equations) [GS94, Chap 6], using three-phase model for unbalanced grids, using voltage-dependent demands [Ari+18], considering voltage-regulated buses and considering transformers and line shunts. However, there is always a trade-off between sufficiently accurate modeling and complexity of the resulting problem. One usually aims at using a model which captures the relevant effects in to sufficient accuracy, but neglects further effects in order to keep the resulting optimization problems tractable. With the set of assumptions given here, we are still able to handle voltage bounds and reactive power.

The AC power flow equations

We model an electrical grid by an undirected graph $G = (\mathcal{N}, \mathcal{E}, \mathcal{Y})$, where $\mathcal{N} = \{1, \dots, N\}$ is the set of buses, $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ is the set of branches and $\mathcal{Y} : \mathcal{E} \to \mathbb{C}^2$, $(k, l) \mapsto (y_{kl}, y^s_{kl})$ is a map assigning a line admittance and a shunt admittance to each branch. We use these the admittances in a π-branch model shown in Figure B.1. In order to derive the relationships between bus powers and all bus voltages, we investigate the relationship between the complex powers flowing into/out of the branch $s_{kl}, s_{lk} \in \mathbb{C}$ and the voltage phasors at the beginning/end of the branch $u_k \in \mathbb{C}$ and $u_l \in \mathbb{C}$ by means of circuit theory.

The complex power flowing from node k to node l is $s_{kl} = u_k i^*_{kl}$, where $i_{kl} \in \mathbb{C}$ is the complex current over the branch (k, l) and z^* denotes the

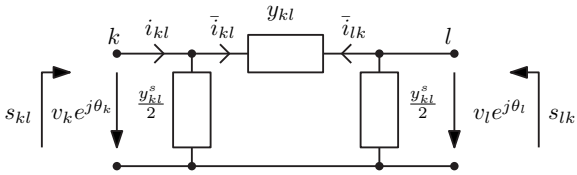


Figure B.1: π-line model of a branch.

complex-conjugate of $z \in \mathbb{C}$. Furthermore we have $\bar{i}_{kl} = (u_k - u_l)y_{kl}$ and $\bar{i}_{kl} = i_{kl} - u_k \frac{y^s_{kl}}{2}$. Thus,

$$s_{kl} = u_k \left((u_k^* - u_l^*) y_{kl}^* + u_k^* \frac{y_{kl}^{s*}}{2} \right). \tag{B.1}$$

By the law of energy conservation, the complex (net) power at node k, s_k has to be the sum of all powers flowing into connected transmission lines yielding

$$s_k = \sum_{k=1}^{N} u_k \left((u_k^* - u_l^*) y_{kl}^* + u_k^* \frac{y_{kl}^{s*}}{2} \right) \quad \text{for all} \quad k \in \mathcal{N}. \tag{B.2}$$

By defining a *bus admittance matrix*

$$[Y_{kl}] := \begin{cases} \sum_{m=1}^{N} y_{km} + \frac{y^s_{km}}{2}, & \text{if} \quad k = l, \\ -y_{kl}, & \text{if} \quad k \neq l, \end{cases}$$

we can write (B.2) compactly as

$$s = \operatorname{diag}(v)\, Y^* v^*. \tag{B.3}$$

Here, $v = (u_k)_{k=1,\dots,N} \in \mathbb{C}^N$ is the vector of all nodal voltages, $s = (s_k)_{k=1,\dots,N} \in \mathbb{C}^N$ denotes the vector of all complex power injections and we define $\operatorname{diag}(z) \in \mathbb{C}^{\mathrm{n_z} \times \mathrm{n_z}}$ as a diagonal matrix with the entries of $z \in \mathbb{C}^{n_z}$ on the main diagonal. If multiple components such as loads, generators and storages are connected to one node we have

$$s_k = s_k^{\mathrm{g}} - s_k^{\mathrm{d}} - s_k^{\mathrm{s}} \quad \text{for all} \quad k \in \mathcal{N}, \tag{B.4}$$

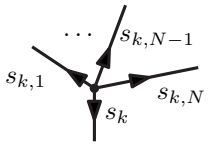


Figure B.2: Apparent power balance at node k.

where $s_k^{\mathrm{g}} \in \mathbb{C}$ denotes apparent power of a generator, $s_k^{\mathrm{d}} \in \mathbb{C}$ denotes the apparent power demand and $s_k^{\mathrm{s}} \in \mathbb{C}$ denotes the apparent power of a storage all connected to bus k. If there is no generator/demand/storage at a node, we set the corresponding power to zero.

As numerical algorithms are typically designed for real-valued spaces, we have to transform (B.3) to the real domain. Here, we have different possibilities: we can represent the complex voltages u_k at all nodes $k = 1, \dots, N$ in cartesian coordinates $u_k = v_k^{\mathrm{re}} + j v_k^{\mathrm{im}}$ or in polar coordinates $u_k = v_k e^{j\theta_k}$, where v_k is the voltage magnitude and θ_k is the voltage angle at node k. The same choice we have for the complex line admittances y_{kl}. The choice of these coordinates leads to different forms of the power flow equations—for details we refer to the excellent survey [FR16]. Here we use the combination of polar coordinates for v and s, and rectangular coordinates for $Y = G + jB$, which is the most popular formulation for OPF. By doing so and by taking real and imaginary parts of (B.3) we get

$$\mathrm{Re}(s_k) = p_k = v_k \sum_{k=1}^{N} v_l (G_{kl} \cos(\theta_{kl}) + B_{kl} \sin(\theta_{kl})), \tag{B.5a}$$

$$\mathrm{Im}(s_k) = q_k = v_k \sum_{k=1}^{N} v_l (G_{kl} \sin(\theta_{kl}) - B_{kl} \cos(\theta_{kl})), \tag{B.5b}$$

for all nodes for all nodes $k \in \mathcal{N}$, where $\theta_{kl} := \theta_k - \theta_l$ and where p_k is called the *active* power and q_k is called the *reactive* power at node k. Note that p_k and q_k are *net powers* (i.e. the residual between generation and demand at node k); substituting p^k and q^k in (B.5) with the corresponding real/imaginary part of equation (B.4) yields power flow equations in terms of component powers.

B.2 Power flow analysis

Recall that classically the only quantities which we can directly influence in power systems are active and reactive power injections from generators p_k^{g} and q_k^{g}. But how to compute the voltage angles θ_k and voltage magnitudes v_k for given power injection and demand? This question is subject of so-called *power flow* analysis.

As the power flow equations (B.5) are usually not algebraically solvable for (v, θ), we have to use iterative methods. Let us write the power flow equations (B.5) with (B.4) as

$$F_k(\chi_k) := \begin{pmatrix} p_k^{\mathrm{g}} - p_k^{\mathrm{d}} - p_k^{\mathrm{s}} - v_k \sum_{k=1}^{N} v_l (G_{kl}\cos(\theta_{kl}) + B_{kl}\sin(\theta_{kl})) \\ q_k^{\mathrm{g}} - q_k^{\mathrm{d}} - q_k^{\mathrm{s}} - v_k \sum_{k=1}^{N} v_l (G_{kl}\sin(\theta_{kl}) - B_{kl}\cos(\theta_{kl})) \end{pmatrix} = 0$$

for all nodes except the first node $k = 2, \ldots, N$ with $\chi_k := (v_k, \theta_k,)^\top$. Let us assume that the variables $p_k^{\mathrm{g}}, q_k^{\mathrm{g}}, p_k^{\mathrm{d}}, p_k^{\mathrm{s}}, q_k^{\mathrm{d}}$ and q_k^{s} are fixed and given here. This yields an implicit function $\tilde{F}(x) := (F_k(\chi_k))_{k=2,\ldots,N}$, where $x := (\chi_k)_{k=2,\ldots,N}$.

Bus one is not included in $\tilde{F}$ and has a special role here: technically speaking, this bus has to compensate the power demand (active and reactive power) which is not covered by the other generators. Physically this follows from the law of energy conservation. Mathematically, this follows from the fact that the last power flow equation can be expressed as a linear combination of the other power flow equations (if no shunts are present as we assume here) implying that one of the power flow equations is "redundant". This leads to the situation that we can not apply the implicit function theorem for locally inverting $\tilde{F}$ (because of the singularity of ∇F). Hence, also iterative methods such as Newton-type methods would fail. The common approach for tackling this issue is to introduce a so-called *slack-bus* at a bus with a large generator (typically bus one), where instead of fixing the net powers p_1 and q_1, we fix the voltage angle θ_1 and voltage magnitude v_1. Hence, the active and reactive powers become "free variables" at this node in the sense that they are a result

rather than a precondition of our computation.[2] Thus, the extended implicit function reads

$$F(x) := \left(v_1 - v_1^{\text{ref}}, \quad \theta_1 - \theta_1^{\text{ref}}, \quad (F_k(\chi_k))_{k=2,\dots,N}\right) \stackrel{!}{=} 0, \tag{B.6}$$

with $x := \left(p_1,\ q_1,\ (v_k, \theta_k)_{k=2,\dots,N}\right)$ where v_1^{ref} and θ_1^{ref} are constant and given reference values. Note that the components of F become linearly independent and $\nabla F(x)$ becomes invertible. The above equations can now be solved by standard Newton-type methods from Section 2.1.2, which is then called a *power-flow study*.

Remark 21 (Different bus types) Note that we assume here for simplicity that we only have two bus types: "pq-buses", where active and reactive power is given and a slack bus, where the voltage angle and magnitude are given. In many works, also a third bus type is considered ("pv-buses"), where active power and the voltage magnitude are given and the voltage angle and the reactive power are unknown. This comes from the fact that some generators are equipped with voltage regulators, which keep the voltage magnitude at a certain value by injecting reactive power. We leave this out for simplicity here as our goal is to outline main idea of power flow computations and we refer to [GS94; WW13] for further details.

B.3 Optimal power flow

The classical goal of *optimal power flow* is to compute (in a certain sense) optimal set-points for all controllable devices in a grid such that a certain cost function is minimized. Thereby, technical and physical limitations such as power generation limits of the power flow equations (B.5) are considered.

[2] This assumption is technically sound if a large generator is connected to the first bus which can typically be ensured by an appropriate numbering of the buses.

In many cases, the objective function in OPF is the sum of the cost function for active power generation of all generators [Zhu15; FR16]

$$f(x) := \sum_{k=1}^{N} f_k^{\mathrm{g}}(p_k^{\mathrm{g}}),$$

where we have $x := (\chi_k)_{k=1,\dots,N}$ with $\chi_k := (v_k,\, \theta_k,\, p_k^{\mathrm{g}},\, q_k^{\mathrm{g}})^\top$ and $p_k^{\mathrm{d}}, p_k^{\mathrm{s}}, q_k^{\mathrm{d}}$, q_k^{s} fixed and given. In contrast to the power flow problem from the previous subsection, all generator powers are "free" variables now, i.e. we would like to compute them optimally. The individual generator cost functions are often chosen as quadratic functions

$$f_k^{\mathrm{g}}(p_k^{\mathrm{g}}) := a_k\,(p_k^{\mathrm{g}})^2 + b_k\,p_k^{\mathrm{g}} + c_k$$

for all $k = 1, \dots, N$ with $f_k^{\mathrm{g}} \equiv 0$ for non-generator buses. Note that the generator cost generally does *not* depend on the reactive power injection q_k^{g}. Reactive power can be seen as a power to charge and discharge transmission lines and inductive/capacitive loads—thus this energy oscillates between consumer and is thus *not* related to any generation cost (except for slightly changing grid losses).

Power generators are usually subject to active and reactive power limitations. These limitations come from operation limits of the power generation units and for the reactive power mainly from current limitations in the generator stator winding. Moreover, voltage magnitudes have to stay withing certain bounds to ensure proper functioning of the connected devices and to avoid damaging the electrical insulation. The current in transmission lines is also constrained by thermal limits. This is often expressed (although not entirely correct)[3] by limits on the magnitude of the apparent power over lines

$$|s_{kl}(x)| := \sqrt{p_{kl}(x)^2 + q_{kl}(x)^2}, \tag{B.7}$$

[3] To be more precise, one would have to limit the magnitude of the current over the transmission line $|i_{kl}|$. As voltage magnitudes typically deviate only slightly from the nominal voltage, the approximation by an apparent power limit is sufficiently tight in most cases.

where $p_{kl} = \mathrm{Re}(s_{kl})$ and $q_{kl} = \mathrm{Im}(s_{kl})$ with s_{kl} from (B.1). Stacking the apparent powers for all transmission lines yields

$$|S|^2(x) := \left(|s_{kl}(x)|^2\right)_{\mathcal{E}}.$$

Combining the above yields a (single-stage) *optimal power flow problem*

$$\min_x \ f(x) \tag{B.8a}$$

$$\text{subject to} \quad F(x) = 0, \tag{B.8b}$$

$$|S|^2(x) \leq \bar{s}^2 \tag{B.8c}$$

$$\underline{x} \leq x \leq \bar{x}, \tag{B.8d}$$

where $\underline{x}$ and $\bar{x}$ collect lower/upper bounds on power injections and voltage magnitudes; $\bar{s}^2$ collects the upper limits on the apparent power over branches.

Remark 22 (OPF variants) Note that there exist many variants of OPF, differing for example in the underlying grid model (DC model vs. AC model, polar coordinates vs. rectangular coordinates, ...) [Zhu15; FR16; MH19], considering uncertain power injection and/or demands [FGM18; M∓17; Müh+19; BCH14] or extensions considering component failure, which is called security constrained OPF [MPG87; Cap+11]. Trying to give an exhaustive overview is beyond the scope of the present thesis and we refer for example to [FR16; FSR12; Zhu15; Cap+11; Cap16] for details. However, the majority of these problem formulations are extensions of the "standard OPF problem" presented here and we see the here presented formulation as a basis for further studies.

B.4 Distributed reformulation

In order to apply distributed optimization algorithms, the above problems have to be in affinely-coupled separable form (2.12). In this section we describe one way of doing so.

In (2.12), any nonlinear equality constraint such as the AC power flow equations (B.5) have to be *local constraints* collected either in g_i or h_i, such that the coupling between the subsystems is solely affine and in form of (2.12d). There

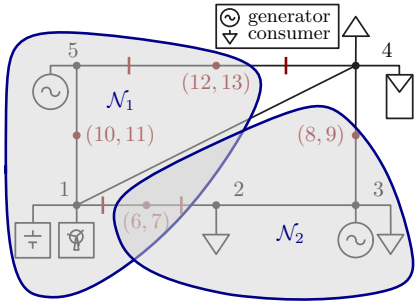


Figure B.3: Example grid with partitioning.

are multiple ways for achieving affine coupling: here we introduce auxiliary nodes in the middle of transmission lines interconnecting two subsystems and coupling active/reactive power and the voltage phasor at these nodes. Other works such as [Ers15] use coupling in the voltage angles and magnitudes only. Another option is, to introduce variable copies of the bus states connected to border lines and to couple the original states again with their copies by means of (2.12d) without introducing any additional buses [HA09]. Our formulation has the advantage, that no "internal information" from the interior of the subsystems has to be exchanged and that two neighbored subsystems share the cost for the line losses of the coupling line. However, other methods may benefit from a reformulated problem with a smaller dimension due to the absence of auxiliary buses.

Let us assume that a grid partitioning is given (for example by the control the area of responsibility of the corresponding DSOs/TSOs or by borders between different grid levels). This means that we have a set of subsystems $\mathcal{R} = \{1, \dots, R\}$, each equipped with a corresponding subset of buses $\mathcal{N}_i \subset \mathcal{N}$ where these subsets are disjoint $\mathcal{N}_i \cap \mathcal{N}_j = \emptyset$ for all $i, j \in \mathcal{R}$ with $i \neq j$. Let us consider a concrete example: Figure B.3 shows a partitioning for the example 5-bus system from [LB10]. We decompose the node set $\mathcal{N} = \{1, \dots, 5\}$ into three regions $\mathcal{R} = \{1, 2, 3\}$ with $\mathcal{N}_1 = \{1, 5\}$, $\mathcal{N}_2 = \{2, 3\}$ and $\mathcal{N}_3 = \{4\}$ fulfilling the above assumptions.

Splitting lines between subsystems

Consider the π-branch model from Figure B.1 shown in Figure B.4, where we assume that the line between node k and node l is a transmission line connecting two subsystems. We introduce two auxiliary buses m and n which are connected to the corresponding buses k and l in the subsystems—thus, we replace the original π-branch model by two π-branch models in series.[4] Regarding the line parameters, we double the series admittance of the original connection y_{kl} and divide the shunt admittance y_{kl}^{s} by two. With that we get an auxiliary buses m and n, which we collect in an auxiliary node set $\mathcal{N}^A$ and an auxiliary node pair (m, n) which we collect in a set $\mathcal{A} \subseteq \mathcal{N}^A \times \mathcal{N}^A$ and new transmission lines $\mathcal{E}^A \subset \mathcal{N} \times \mathcal{N}^A$. To obtain equivalence to the original problem, we require

$$u_m = u_n \quad \text{and} \quad s_{k,m} = -s_{n,l} \quad \text{for all } (m, n) \in \mathcal{A}. \tag{B.9}$$

Taking real part and imaginary part of (B.9) yields

$$v_m = v_n \qquad p_{k,m} = -p_{n,l} \tag{B.10a}$$

$$\theta_m = \theta_n, \qquad q_{k,m} = -q_{n,l}. \tag{B.10b}$$

for all $(m, n) \in \mathcal{A}$ resulting in an *affine coupling* in form of (2.12d).

In order to simplify presentation, we will assume from now on that the auxiliary buses are already included in $\mathcal{N}$ and $\mathcal{N}_i$ respectively. More precisely, this means that we introduce extended bus sets $\mathcal{N}^E = \mathcal{N} \cup \mathcal{N}^A$ and $\mathcal{N}_i^E = \mathcal{N}_i \cup \{m \in \mathcal{N}^A \mid \text{there exists } (k, m) \in \mathcal{E}^A\}$. In the following we use the symbol $\mathcal{N}$ for the extended bus set $\mathcal{N}^E$ for simplicity.

[4] Note that this reformulation is equivalent to the original model only if $y_{k,l}^{\mathrm{s}} = 0$. However, from a physical perspective, a series connection of multiple π-elements can in general more accurately describe the physical behavior of the branch as it allows for a more accurate approximation of the hyperbolic equations of transmission lines [Sch17, Ch. 10.3].

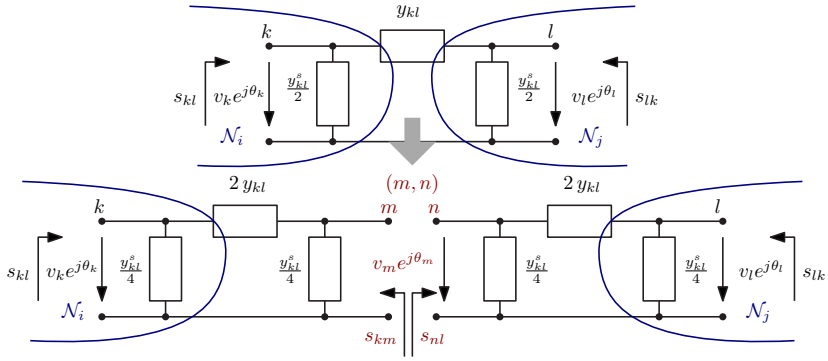


Figure B.4: Decomposition of a π-line model.

The reformulated problem

With the above, we can construct the power flow equations for each region $i \in \mathcal{R}$ as

$$F_i(x_i) := (F_k(\chi_k))_{k \in \mathcal{N}_i},$$

where $x_i := (\chi_k)_{k \in \mathcal{N}_i}$. Similarly, we have for the line limits and box constraints

$$|S|_i^2(x) := \left(|s_{k,l}|(x)^2\right)_{\mathcal{N}_i \times \mathcal{N}_i} \quad \text{and} \quad \underline{x}_i \le x_i \le \bar{x}_i$$

for all $i \in \mathcal{R}$. Furthermore, we define for all $i \in \mathcal{R}$

$$f_i(x_i) := \sum_{k \in \mathcal{N}_i} f_k^{\mathrm{g}}(p_k^{\mathrm{g}}).$$

This yields an OPF problem in form of (2.12)

$$\min_{x_1,\dots,x_R} \sum_{i\in\mathcal{R}} f_i(x_i) \tag{B.11a}$$

$$\text{subject to} \quad F_i(x_i) = 0 \qquad \text{for all} \quad i \in \mathcal{R}, \tag{B.11b}$$

$$|S|_i^2(x_i) \le \bar{s}_i^2 \qquad \text{for all} \quad i \in \mathcal{R}, \tag{B.11c}$$

$$\underline{x}_i \le x_i \le \bar{x}_i \qquad \text{for all} \quad i \in \mathcal{R}, \tag{B.11d}$$

$$\sum_{i\in\mathcal{R}} A_i\, x_i = 0, \tag{B.11e}$$

where the matrices $A_i \in \mathbb{R}^{4|\mathcal{N}^E|\times 4|\mathcal{N}_i|}$ are constructed such that the coupling constraints (B.10) are satisfied.

C Insights on ADMM

C.1 Modulus switching in ADMM

As we have seen in Section 5.5, d-ADMM seems to be able to reach only a limited accuracy. In this section we will investigate this behavior on a small-scale example. It turns out that the accuracy is not limited, but the convergence is so slow that is seems like ADMM is not making any progress. Moreover, we show that by choosing a different ρ, one can achieve a higher accuracy but then the overall convergence to that accuracy is slow. So there seems to be a trade-off between the achievable accuracy and the convergence speed.

Consider an unconstrained strongly convex QP with objective function $f(x) = \frac{1}{2}x^\top Hx - h^\top x$, where $H = \sum_{i\in\mathcal{R}} H_i \succ 0$ and $h = \sum_{i\in\mathcal{R}} h_i$ with ADMM similar to the inner problems in bi-level ALADIN from Section 4.2.1. Let

$$H_1 = \begin{pmatrix} a & c & c \\ c & d & 0 \\ c & 0 & 0 \end{pmatrix}, \quad H_2 = \begin{pmatrix} a & c+d & c \\ c+d & b & d \\ c & d & 0 \end{pmatrix}, \quad H_3 = \begin{pmatrix} 0 & d & 0 \\ d & b & d \\ 0 & d & 0 \end{pmatrix},$$

and $h_1 = (a \quad 0 \quad 0)^\top$, $h_2 = (a \quad b \quad 0)^\top$, $h_3 = (0 \quad b \quad 0)^\top$ with $a = 10^{-4}$, $b = 5$, $c = 10^{-6}$ and $d = 10^{-2}$. Then we have a condition number $\text{cond}(H) = 6.25 \cdot 10^4$ which should not pose too severe difficulties.

Figure C.1 shows the resulting convergence plots for d-ADMM for different values of ρ. One can clearly see that the accuracy in the primal and dual variables seems limited for this problem to round about 10^{-5} for $\rho = 10^{-4}$. By choosing a larger ρ of $5 \cdot 10^{-3}$ or 10^{-2}, one can achieve higher accuracies, but in this case one needs several thousand iterations to reach an accuracy of 10^{-5} in the primal variables. In practice, this is often not an option and especially not in bi-level ALADIN as then the overall communication would be very high. So in practice, tuning of ρ is typically done in a way such that one

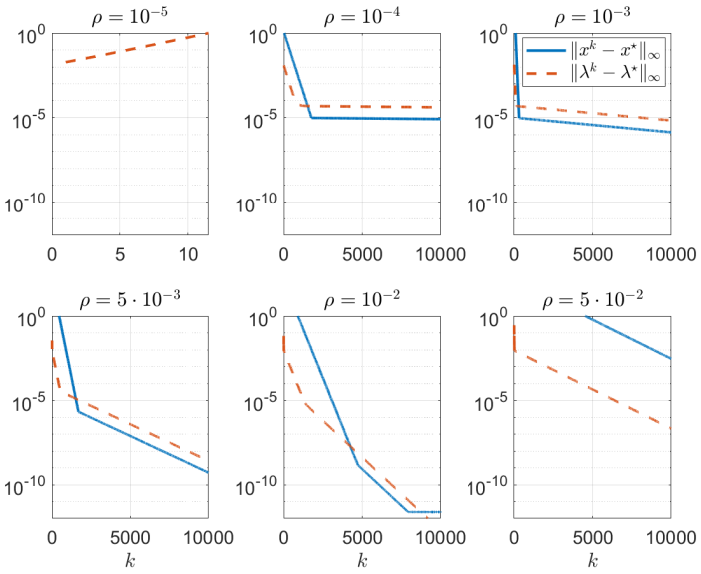


Figure C.1: ADMM for solving a unconstrained convex QP.

gets fast convergence in the primal variables in the beginning of the ADMM iterations. Hence, for the here-considered example one would probably choose a ρ around 10^{-3} since on can then reach an accuracy of around 10^{-4} after several tens/hundreds of iterations. By doing so one also essentially limits the accuracy of ADMM to that level. Note that for small $\rho = 10^{-5}$, ADMM diverges as the individual f_is are unbounded below since the matrices H_i are indefinite.

From our example one can observe another interesting effect: ADMM seems to switch it's linear convergence modulus while iterating. Especially in case of $\rho = 10^{-2}$ illustrates one can observe this behavior. A similar behavior is also observed in other works [NOB16; Gol+14]. To the best of our knowledge this effect is not investigated in the literature so far. One can also draw another conclusion from this example: although for strongly convex problems (which our example is), linear convergence has been shown (e.g. [GB16]), the present example (and the examples from the inner problems in bi-level

ALADIN (Section 5.5)) shows that the modulus of convergence for ADMM can be arbitrarily bad. Thus, one has to be very careful with the interpretation of such results—if there considered situation occurs one essentially needs a very large number of iterations for convergence.

Conditions for which slow convergence occurs seems to the best of our knowledge not to exist in the literature. In our particular case the matrices H_i are indefinite leading to non-convex f_is, which might be one reason. However, investigating this in more detail seems to be an important direction of future work.

C.2 Examples: ADMM getting stuck for $\rho \to \infty$

We give an analytic example where one can observe that ADMM gets stuck for $\rho \to \infty$ and a feasible initial point as predicted by Proposition 1. Moreover, we show that ADMM converges to a feasible but not optimal point in case of an infeasible initialization for the same example. In a second example, we show that evaluating primal feasibility only is in general not sufficient for convergence of ADMM, which is sometimes assumed in the power system context [Ers15; GHT17].

Example 4 (ADMM getting stuck for $\rho \to \infty$) Consider the problem $\min_{x,z} \frac{1}{2}x^2$ subject to $x = z$ and $x \geq -1$. The partial augmented Lagrangian with respect to the constraint $x = z$ is $\mathcal{L}_\rho(x, z, \lambda) = \frac{1}{2}x^2 + \lambda(x - z) + \frac{\rho}{2}(x - z)^2$.
Case 1: Consider a feasible z^0, i.e. $z^0 \geq -1$. Applying ADMM (Algorithm 3), minimizing $\mathcal{L}_\rho$ with respect to x yields $x^{k+1} + \lambda^k + \rho(x^{k+1} - z^k) = 0$ and thus $x^{k+1} = \frac{\rho z^k - \lambda^k}{1+\rho} = \frac{z^k - \lambda^k/\rho}{1/\rho+1} \overset{\rho\to\infty}{\to} z^k$. Minimizing with respect to z yields $-\lambda^k - \rho(x^{k+1} - z^{k+1}) = 0$ and thus $z^{k+1} = x^{k+1} + \frac{\lambda^k}{\rho} = \frac{z^k - \lambda^k/\rho}{1/\rho+1} + \frac{\lambda^k}{\rho} \overset{\rho\to\infty}{\to} z^k$. The λ update in step 3) reads $\lambda^{k+1} = \lambda^k + \rho(x^{k+1} - z^{k+1}) = \lambda^k + \rho(x^{k+1} - x^{k+1} - \frac{\lambda^k}{\rho}) = 0$. This show that, as predicted by Proposition 1 ADMM gets stuck for this example at z^k. In case when the initial point is infeasible however, the situation is different as we will see now.
Case 2: Consider an infeasible point $z^k < -1$ and an arbitrary but fixed λ^k. Then, the minimizer of $\mathcal{L}_\rho$ with respect to x, x^{k+1} will be -1 for $\rho \to \infty$ since $x \geq -1$ has to be satisfied. After that iteration, we have similar to the above

$z^{k+1} = x^{k+1} = -1$ for $\rho \to \infty$ and thus $\lambda^{k+1} \to 0$. This show exemplarily, that the behavior of ADMM for large ρ heavily depends on whether or not the initial point is feasible. Note that in case that in case there are also individual constraints for z, the situation might be even more complicated as in this case some kind of "alternating projections" between the constraint sets might occur leading to an oscillatory behavior. Moreover, note that a similar situation occurs if one increases ρ to very large values as ADMM proceeds as for example done in [Ers15; GHT17].

Example 5 (Primal feasibility is not sufficient for termination) Consider example 4 again neglecting the constraint $x \geq -1$. Let us furthermore assume that we use ADMM with primal feasibility ($|x - z| < \epsilon$) as the only termination criterion. From Example 4 we know that $|x^{k+1} - z^{k+1}| = |\lambda^k/\rho|$. Thus shows that if we choose a ρ large enough such that $|\lambda^k/\rho| < \epsilon$, ADMM will imediately terminate although even though (x^{k+1}, z^{k+1}) is not optimal.

D Distributed Optimization and Computer Science

Distributed optimization and computer science are strongly connected. One reason for this is that distributed optimization has many promising applications in computer science beyond a multi-agent setting. One can identify applications in at least three branches:

- in supercomputing [CZ97; BT89; Don+03];
- in network optimization [BT89; San+19; Chi+07];
- in machine learning, specifically
  - in image reconstruction [Yan+16; ZK14; XY13];
  - and in support vector machines [FCG10; Boy+11].

In this thesis we present a particular SVM example in Chapter 2. An additional logistic regression example from machine learning is included in the example set of ALADIN-α available under [Eng20a].

Also from an historical perspective, many pioneers of distributed optimization are closely connected to computer science. John von Neumann's alternating projection method [Neu49] for example is closely related to ADMM; or John Tsitsiklis with his work on distributed and asynchronous algorithms [Tsi94; BT89]. These roots can also be observed in terminology such as nonlinear/integer/quadratic *programming*.

Moreover, interconnections from systems and control and optimization over networks/theoretical informatics exist. The Bellmann-Ford algorithm for example was invented by one of the most famous people working in systems and control and pioneers of so-called dynamic programming: Richard Bellmann [Bel58]. Even Dijkstra's algorithm can be cast within the framework

of dynamic programming which is often considered to be a "control method" [Sni06]. Hence, distributed optimization can be seen as an interdisciplinary field connecting computer science, applied mathematics and systems and control.

Distributed optimization vs. parallel computing

In computer science, a distinction between *parallel computing* and *distributed computing* is often made. Parallel computing usually means that computation is done on multiple processors with one shared memory. Distributed computing typically refers to the case, where computers are connected via a network with a distributed memory and operate via message-passing [BT89; Don+03]. This work focuses mainly on distributed computing altough the developed algorithms can also be used for parallel computing.

Although distributed optimization techniques are interesting for parallel computation, the requirements are typically slightly different. Whereas in parallel computing partitioning is usually done for load balancing—i.e. the problem size handled by each processor is approximately the same for all processors—in our setting the partitioning is usually given a priori by the subsystem boundaries. Furthermore, in parallel computing, communication is usually not of such a big concern because of the fast and high-throughput shared memory.

Distributed optimization vs. optimization over networks

The problems considered in this thesis are different from classical optimization problems in theoretical informatics defined over graphs such as shortest-path problems:[1] here we consider continuous optimization problems without integers and with graphs only as a side-topic.

In contrast to network optimization, in our setting complexity is mostly "in the nodes" rather than "in the network" in the sense that in network flow the nodes have typically only one decision variable and the edges are more important. In

[1] Although certain interconnections may exist. The duality theory from Chapter 2 can for example be fruitfully used in context of network flow problems [Roc98; Ber98].

this thesis, we consider problems, where each node is a (complex) optimization problem on its own, for example itself being a dynamical system or an area of a power grid. Methods from network optimization are not straight-forward to use in this setting. An additional difference to classical optimization over networks is that here the edges in general come with underlying physical laws. Although electrical grids can be seen as a graph with complex-valued weights [Lei+15], for the AC grid model the nodes are coupled via nonlinear equations making classical algorithms from theoretical informatics hard to use.

References

[Bot+20] M. Botkin-Levy, A. Engelmann, T. Mühlpfordt, T. Faulwasser, and M. Almassalkhi. “Distributed Control of Charging for Electric Vehicle Fleets under Dynamic Transformer Ratings”. In: *submitted to IEEE Transactions on Control Systems Technology* (July 2020). arXiv: 2007.10304.

[Boy+20] S. Boyd, N. Parikh, E. Chu, P. Borja, and J. Eckstein. *MATLAB Scripts for Alternating Direction Method of Multipliers*. July 2020. URL: https://web.stanford.edu/~boyd/papers/admm/ (visited on 07/27/2020).

[Du+20] X. Du, A. Engelmann, Y. Jiang, T. Faulwasser, and B. Houska. “Optimal Experiment Design for AC Power Systems Admittance Estimation”. In: *Proceedings of the 21th IFAC World Congress, Berlin*. 2020.

[ene20] ene’t GmbH. *Karte Der Stromnetzbetreiber, Niederspannung, Deutschland, Januar 2020*. July 2020. URL: https://www.enet.eu/media/portfolio/karten/Karte-der-Stromnetzbetreiber-Niederspannung-2020.jpg, License:CreativeCommonsBY-SA3.0DE,figurechanged (visited on 07/16/2020).

[Eng+20] A. Engelmann, H. Benner, R. Ou, Y. Jiang, B. Houska, and T. Faulwasser. “ALADIN-Alpha—An Open-Source MATLAB Toolbox for Distributed Non-Convex Optimization”. In: *submitted to Optimal Control Applications and Methods* (2020). arXiv: 2006.01866.

[Eng20a] A. Engelmann. *ALADIN-ALPHA*. 2020. URL: https://alexe15.github.io/ALADIN.m/ (visited on 07/27/2020).

[Eng20b] A. Engelmann. *Options - ALADIN-ALPHA*. 2020. URL: https://alexe15.github.io/ALADIN.m/options/ (visited on 07/27/2020).

[Eng+20] A. Engelmann, Y. Jiang, B. Houska, and T. Faulwasser. “Decomposition of Non-Convex Optimization via Bi-Level Distributed ALADIN”. In: *IEEE Transactions on Control of Network Systems* (2020). DOI: 10.1109/TCNS.2020.3005079.

[GGC20] W. Gao, D. Goldfarb, and F. E. Curtis. “ADMM for Multiaffine Constrained Optimization”. In: *Optimization Methods and Software* 35.2 (2020), pp. 257–303.

[Guo20] J. Guo. *Guojunyao419/OPF-ADMM*. Feb. 2020. URL: https://github.com/guojunyao419/OPF-ADMM (visited on 07/27/2020).

[Jia+20] Y. Jiang, P. Sauerteig, B. Houska, and K. Worthmann. "Distributed Optimization Using ALADIN for MPC in Smart Grids". In: *arXiv:2004.01522* (Apr. 2020). arXiv: 2004.01522.

[KKS20a] J. Kardoš, D. Kourounis, and O. Schenk. "Structure-Exploiting Interior Point Methods". In: *Parallel Algorithms in Computational Science and Engineering*. Ed. by A. Grama and A. H. Sameh. Modeling and Simulation in Science, Engineering and Technology. Cham: Springer International Publishing, 2020, pp. 63–93. DOI: 10.1007/978-3-030-43736-7_3.

[KKS20b] J. Kardoš, D. Kourounis, and O. Schenk. "Two-Level Parallel Augmented Schur Complement Interior-Point Algorithms for the Solution of Security Constrained Optimal Power Flow Problems". In: *IEEE Transactions on Power Systems* 35.2 (Mar. 2020), pp. 1340–1350. DOI: 10.1109/TPWRS.2019.2942964.

[LT20] D. Liu and Q. Tran-Dinh. "An Inexact Interior-Point Lagrangian Decomposition Algorithm with Inexact Oracles". In: *Journal of Optimization Theory and Applications* 185.3 (June 2020), pp. 903–926. DOI: 10.1007/s10957-020-01680-3.

[Mao+20] X. Mao, K. Yuan, Y. Hu, Y. Gu, A. H. Sayed, and W. Yin. "Walkman: A Communication-Efficient Random-Walk Algorithm for Decentralized Optimization". In: *IEEE Transactions on Signal Processing* (2020), pp. 1–1. DOI: 10.1109/TSP.2020.2983167.

[Mot+20] J. Mota, J. Xavier, P. Aguiar, and M. Püschel. *D-ADMM*. July 2020. URL: http://users.isr.ist.utl.pt/~jmota/DADMM/ (visited on 07/27/2020).

[Pu+20] S. Pu, W. Shi, J. Xu, and A. Nedic. "Push-Pull Gradient Methods for Distributed Optimization in Networks". In: *IEEE Transactions on Automatic Control* (2020), pp. 1–1. DOI: 10.1109/TAC.2020.2972824.

[Ste+20] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. "OSQP: An Operator Splitting Solver for Quadratic Programs". In: *Mathematical Programming Computation* (Feb. 2020). DOI: 10.1007/s12532-020-00179-2.

[Su+20] J. Su, Y. Jiang, A. Bitlislioglu, C. N. Jones, and B. Houska. "Distributed Multi-Building Coordination for Demand Response". In: *Proceeding of the 21st IFAC World Congress*. Berlin, 2020.

[And+19] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. "CasADi: A Software Framework for Nonlinear Optimization and Optimal Control". In: *Mathematical Programming Computation* 11.1 (Mar. 2019), pp. 1–36. DOI: 10.1007/s12532-018-0139-4.

[BG19] A. Bestler and K. Graichen. "Distributed Model Predictive Control for Continuous-Time Nonlinear Systems Based on Suboptimal ADMM". In: *Optimal Control Applications and Methods* 40.1 (2019), pp. 1–23. DOI: 10.1002/oca.2459.

[Bun19] R. Bundesnetzagentur. *Monitorungbericht 2019*. Tech. rep. Bundesnetzagentur (Federal German Grid Authority), Germany, Bonn, 2019.

[CS19] B. Cui and X. A. Sun. "Solvability of Power Flow Equations Through Existence and Uniqueness of Complex Fixed Point". In: *arXiv:1904.08855* (Apr. 2019). arXiv: 1904.08855.

[Du+19] X. Du, A. Engelmann, Y. Jiang, T. Faulwasser, and B. Houska. "Distributed State Estimation for AC Power Systems Using Gauss-Newton ALADIN". In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. Dec. 2019, pp. 1919–1924. DOI: 10.1109/CDC40024.2019.9028966.

[Eng+19a] A. Engelmann, Y. Jiang, B. Houska, and T. Faulwasser. *Decomposition of non-convex optimization via bi-level distributed ALADIN*. Submitted to the IEEE Transactions on Control of Network Systems, arXiv:1903.11280. 2019.

[Eng+19b] A. Engelmann, Y. Jiang, T. Mühlpfordt, B. Houska, and T. Faulwasser. "Toward Distributed OPF Using ALADIN". In: *IEEE Transactions on Power Systems* 34.1 (Jan. 2019), pp. 584–594. DOI: 10.1109/TPWRS.2018.2867682.

[FE19] T. Faulwasser and A. Engelmann. "Toward Economic NMPC for Multistage AC Optimal Power Flow". In: *Optimal Control Applications and Methods* 41.1 (2019), pp. 107–127. DOI: 10.1002/oca.2487.

[Jia+19] Y. Jiang, D. Kouzoupis, H. Yin, M. Diehl, and B. Houska. "Decentralized Optimization over Tree Graphs". In: *arXiv:1910.09206* (Oct. 2019). arXiv: 1910.09206.

[MSL19] N. Meyer-Huebner, M. Suriyah, and T. Leibfried. "Distributed Optimal Power Flow in Hybrid AC-DC Grids". In: *IEEE Transactions on Power Systems* 34.4 (July 2019), pp. 2937–2946.

[MH19] D. K. Molzahn and I. A. Hiskens. “A Survey of Relaxations and Approximations of the Power Flow Equations”. In: *Foundations and Trends® in Electric Energy Systems* 4.1-2 (Feb. 2019), pp. 1–221. DOI: 10.1561/3100000012.

[Müh+19] T. Mühlpfordt, L. Roald, V. Hagenmeyer, T. Faulwasser, and S. Misra. “Chance-Constrained AC Optimal Power Flow: A Polynomial Chaos Approach”. In: *IEEE Transactions on Power Systems* 34.6 (Nov. 2019), pp. 4806–4816. DOI: 10.1109/TPWRS.2019.2918363.

[Mur+19] A. Murray, T. Faulwasser, V. Hagenmeyer, M. E. Villanueva, and B. Houska. “Partially Distributed Outer Approximation”. In: (2019).

[Pet19] C. G. Petra. “A Memory-Distributed Quasi-Newton Solver for Nonlinear Programming Problems with a Small Number of General Constraints”. In: *Journal of Parallel and Distributed Computing* 133 (Nov. 2019), pp. 337–348. DOI: 10.1016/j.jpdc.2018.10.009.

[PCA19] C. G. Petra, N. Chiang, and M. Anitescu. “A Structured Quasi-Newton Algorithm for Optimizing with Incomplete Hessian Information”. In: *SIAM Journal on Optimization* 29.2 (Jan. 2019), pp. 1048–1075. DOI: 10.1137/18M1167942.

[QL19] G. Qu and N. Li. “Accelerated Distributed Nesterov Gradient Descent”. In: *IEEE Transactions on Automatic Control* (2019), pp. 1–1. DOI: 10.1109/TAC.2019.2937496.

[Rog+19] A. Rogozin, C. Uribe, A. Gasnikov, N. Malkovskii, and A. Nedich. “Optimal Distributed Convex Optimization on Slowly Time-Varying Graphs”. In: *IEEE Transactions on Control of Network Systems* (2019), pp. 1–1. DOI: 10.1109/TCNS.2019.2949439.

[ST19a] S. Sabach and M. Teboulle. “Lagrangian Methods for Composite Optimization”. In: *Handbook of Numerical Analysis*. Vol. 20. Elsevier, 2019, pp. 401–436. DOI: 10.1016/bs.hna.2019.04.002.

[San+19] P. Sanders, K. Mehlhorn, M. Dietzfelbinger, and R. Dementiev. *Sequential and Parallel Algorithms and Data Structures: The Basic Toolbox*. Springer International Publishing, 2019. DOI: 10.1007/978-3-030-25209-0.

[Sul+19] J. Sulam, A. Aberdam, A. Beck, and M. Elad. “On Multi-Layer Basis Pursuit, Efficient Algorithms and Convolutional Neural Networks”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019), pp. 1–1. DOI: 10.1109/TPAMI.2019.2904255.

[ST19b] T. Sun and Q. Tran-Dinh. “Generalized Self-Concordant Functions: A Recipe for Newton-Type Methods”. In: *Mathematical Programming* 178.1 (Nov. 2019), pp. 145–213. DOI: 10.1007/s10107-018-1282-4.

[TBJ19] R. Tutunov, H. Bou-Ammar, and A. Jadbabaie. “Distributed Newton Method for Large-Scale Consensus Optimization”. In: *IEEE Transactions on Automatic Control* 64.10 (Oct. 2019), pp. 3983–3994. DOI: 10.1109/TAC.2019.2907711.

[WYZ19] Y. Wang, W. Yin, and J. Zeng. “Global Convergence of ADMM in Nonconvex Nonsmooth Optimization”. In: *Journal of Scientific Computing* 78.1 (Jan. 2019), pp. 29–63. DOI: 10.1007/s10915-018-0757-z.

[Yan+19] T. Yang, X. Yi, J. Wu, Y. Yuan, D. Wu, Z. Meng, Y. Hong, H. Wang, Z. Lin, and K. H. Johansson. “A Survey of Distributed Optimization”. In: *Annual Reviews in Control* 47 (Jan. 2019), pp. 278–305. DOI: 10.1016/j.arcontrol.2019.05.006.

[Zwi+19] P. Zwickel, A. Engelmann, L. Gröll, V. Hagenmeyer, D. Sauer, and T. Faulwasser. “A Comparison of Economic MPC Formulations for Thermal Building Control”. In: *2019 IEEE PES Innovative Smart Grid Technologies Europe*. Sept. 2019, pp. 1–5. DOI: 10.1109/ISGTEurope.2019.8905593.

[Ari+18] A. Arif, Z. Wang, J. Wang, B. Mather, H. Bashualdo, and D. Zhao. “Load Modeling—A Review”. In: *IEEE Transactions on Smart Grid* 9.6 (Nov. 2018), pp. 5986–5999. DOI: 10.1109/TSG.2017.2700436.

[Bec+18] A. Beck, Y. Beck, Y. Levron, A. Shtof, and L. Tetruashvili. “Globally Solving a Class of Optimal Power Flow Problems in Radial Networks by Tree Reduction”. In: *Journal of Global Optimization* 72.3 (Nov. 2018), pp. 373–402. DOI: 10.1007/s10898-018-0652-z.

[BST18] J. Bolte, S. Sabach, and M. Teboulle. “Nonconvex Lagrangian-Based Optimization: Monitoring Schemes and Global Convergence”. In: *Mathematics of Operations Research* 43.4 (Apr. 2018), pp. 1210–1232. DOI: 10.1287/moor.2017.0900.

[BCN18] L. Bottou, F. E. Curtis, and J. Nocedal. “Optimization Methods for Large-Scale Machine Learning”. In: *SIAM Review* 60.2 (Jan. 2018), pp. 223–311. DOI: 10.1137/16M1080173.

[Dor18] F. Dorfler. “Distributed Consensus-Based Optimization”. Zürich, 2018. URL: http://people.ee.ethz.ch/~floriand/docs/Teaching/ATIC_2018/Optimization_Lecture.pdf.

[EF18] A. Engelmann and T. Faulwasser. “Feasibility vs. Optimality in Distributed AC OPF - A Case Study Considering ADMM and ALADIN”. In: *Proceedings of the Second International Symposium on Energy System Optimization*. 2018. DOI: 10.1007/978-3-030-32157-4_1.

[Eng+18a] A. Engelmann, T. Mühlpfordt, Y. Jiang, B. Houska, and T. Faulwasser. “Distributed Stochastic AC Optimal Power Flow Based on Polynomial Chaos Expansion”. In: *2018 Annual American Control Conference (ACC)*. June 2018, pp. 6188–6193. DOI: 10.23919/ACC.2018.8431090.

[Eng+18b] A. Engelmann, T. Mühlpfordt, Y. Jiang, B. Houska, and T. Faulwasser. “Distributed Stochastic AC Optimal Power Flow based on Polynomial Chaos Expansion”. In: *2018 Annual American Control Conference (ACC)*. 2018, pp. 6188–6193. DOI: 10.23919/ACC.2018.8431090.

[Fau+18] T. Faulwasser, A. Engelmann, T. Mühlpfordt, and V. Hagenmeyer. “Optimal power flow: an introduction to predictive, distributed and stochastic control challenges”. In: *at - Automatisierungstechnik* 66.7 (July 2018), pp. 573–589. DOI: 10.1515/auto-2018-0040.

[FGM18] T. Faulwasser, L. Grüne, and M. A. Müller. “Economic Nonlinear Model Predictive Control”. In: *Foundations and Trends® in Systems and Control* 5.1 (Jan. 2018), pp. 1–98. DOI: 10.1561/2600000014.

[Grö18] L. Gröll. “Klassifikation von Optimierungsproblemen”. In: *at - Automatisierungstechnik* 66.11 (Nov. 2018), pp. 903–927. DOI: 10.1515/auto-2018-0081.

[KFS18] D. Kourounis, A. Fuchs, and O. Schenk. “Toward the Next Generation of Multiperiod Optimal Power Flow Solvers”. In: *IEEE Transactions on Power Systems* 33.4 (July 2018), pp. 4005–4014. DOI: 10.1109/TPWRS.2017.2789187.

[LT18] J. Lu and C. Y. Tang. “A Distributed Algorithm for Solving Positive Definite Linear Equations Over Networks With Membership Dynamics”. In: *IEEE Transactions on Control of Network Systems* 5.1 (2018), pp. 215–227.

[Lu+18] W. Lu, M. Liu, S. Lin, and L. Li. “Fully Decentralized Optimal Power Flow of Multi-Area Interconnected Power Systems Based on Distributed Interior Point Method”. In: *IEEE Trans Power Syst* 33.1 (Jan. 2018), pp. 901–910. DOI: 10.1109/TPWRS.2017.2694860.

[MJ18] M. Maros and J. Jaldén. “PANDA: A Dual Linearly Converging Method for Distributed Optimization Over Time-Varying Undirected Graphs”. In: *2018 IEEE Conference on Decision and Control (CDC)*. Dec. 2018, pp. 6520–6525. DOI: 10.1109/CDC.2018.8619626.

[Mur+18] A. Murray, A. Engelmann, V. Hagenmeyer, and T. Faulwasser. “Hierarchical Distributed Mixed-Integer Optimization for Reactive Power Dispatch”. In: *IFAC-PapersOnLine* 51.28 (2018), pp. 368–373. DOI: 10.1016/j.ifacol.2018.11.730.

[NOW18] A. Nedić, A. Olshevsky, and S. Wei. “Decentralized Consensus Optimization and Resource Allocation”. In: *Large-Scale and Distributed Optimization*. Springer, 2018, pp. 247–287.

[NOR18] A. Nedić, A. Olshevsky, and M. G. Rabbat. “Network Topology and Communication-Computation Tradeoffs in Decentralized Optimization”. In: *Proceedings of the IEEE* 106.5 (May 2018), pp. 953–976. DOI: 10.1109/JPROC.2018.2817461.

[Nes18] Y. Nesterov. *Lectures on Convex Optimization*. Vol. 137. Springer, 2018. DOI: 10.1007/978-3-319-91578-4.

[QL18] G. Qu and N. Li. “Harnessing Smoothness to Accelerate Distributed Optimization”. In: *IEEE Transactions on Control of Network Systems* 5.3 (Sept. 2018), pp. 1245–1260. DOI: 10.1109/TCNS.2017.2698261.

[Sim18] J. W. Simpson-Porco. “A Theory of Solvability for Lossless Power Flow Equations—Part I: Fixed-Point Power Flow”. In: *IEEE Transactions on Control of Network Systems* 5.3 (Sept. 2018), pp. 1361–1372.

[XK18] R. Xin and U. A. Khan. “A Linear Algorithm for Optimization Over Directed Graphs With Geometric Convergence”. In: *IEEE Control Systems Letters* 2.3 (July 2018), pp. 315–320. DOI: 10.1109/LCSYS.2018.2834316.

[ZY18] J. Zeng and W. Yin. “On Nonconvex Decentralized Gradient Descent”. In: *IEEE Transactions on Signal Processing* 66.11 (June 2018), pp. 2834–2848. DOI: 10.1109/TSP.2018.2818081.

[Bec17] A. Beck. *First-Order Methods in Optimization*. Vol. 25. MOS-SIAM Series on Optimization, 2017.

[BJ17] A. Bitlislioğlu and C. N. Jones. “On Coordinated Primal-Dual Interior-Point Methods for Multi-Agent Optimization”. In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. Dec. 2017, pp. 3531–3536. DOI: 10.1109/CDC.2017.8264177.

[BPJ17] A. Bitlislioğlu, I. Pejcic, and C. Jones. “Interior Point Decomposition for Multi-Agent Optimization”. In: *IFAC-PapersOnLine*. 20th IFAC World Congress 50.1 (July 2017), pp. 233–238. DOI: 10.1016/j.ifacol.2017.08.039.

[CZ17] N. Chatzipanagiotis and M. M. Zavlanos. “On the Convergence of a Distributed Augmented Lagrangian Method for Nonconvex Optimization”. In: *IEEE Transactions on Automatic Control* 62.9 (Sept. 2017), pp. 4405–4420. DOI: 10.1109/TAC.2017.2658438.

[Chr+17a] K. Christakou, D.-C. Tomozei, J.-Y. Le Boudec, and M. Paolone. “AC OPF in Radial Distribution Networks – Part I: On the Limits of the Branch Flow Convexification and the Alternating Direction Method of Multipliers”. In: *Electric Power Systems Research* 143 (Feb. 2017), pp. 438–450. DOI: 10.1016/j.epsr.2016.07.030.

[Chr+17b] K. Christakou, D.-C. Tomozei, J.-Y. Le Boudec, and M. Paolone. “AC OPF in Radial Distribution Networks – Part II: An Augmented Lagrangian-Based OPF Algorithm, Distributable via Primal Decomposition”. In: *Electric Power Systems Research* 150 (Sept. 2017), pp. 24–35. DOI: 10.1016/j.epsr.2017.04.028.

[Eng+17] A. Engelmann, T. Mühlpfordt, Y. Jiang, B. Houska, and T. Faulwasser. “Distributed AC Optimal Power Flow Using ALADIN”. In: *IFAC-PapersOnLine*. Vol. 50. 20th IFAC World Congress. July 2017, pp. 5536–5541. DOI: 10.1016/j.ifacol.2017.08.1095.

[Fer+17] H. J. Ferreau, S. Almér, R. Verschueren, M. Diehl, D. Frick, A. Domahidi, J. L. Jerez, G. Stathopoulos, and C. Jones. “Embedded Optimization Methods for Industrial Automatic Control”. In: *IFAC-PapersOnLine*. 20th IFAC World Congress 50.1 (July 2017), pp. 13194–13209. DOI: 10.1016/j.ifacol.2017.08.1946.

[GB17] P. Giselsson and S. Boyd. “Linear Convergence and Metric Selection for Douglas-Rachford Splitting and ADMM”. In: *IEEE Transactions on Automatic Control* 62.2 (Feb. 2017), pp. 532–544. DOI: 10.1109/TAC.2016.2564160.

[GOY17] R. Glowinski, S. J. Osher, and W. Yin. *Splitting Methods in Communication, Imaging, Science, and Engineering*. Springer, 2017. DOI: 10.1007/978-3-319-41589-5.

[GHT17] J. Guo, G. Hug, and O. K. Tonguz. “A Case for Nonconvex Distributed Optimization in Large-Scale Power Systems”. In: *IEEE Transactions on Power Systems* 32.5 (Sept. 2017), pp. 3842–3851. DOI: 10.1109/TPWRS.2016.2636811.

[HL17] M. Hong and Z. Luo. “On the Linear Convergence of the Alternating Direction Method of Multipliers”. In: *Mathematical Programming* 162.1-2 (2017), pp. 165–199.

[HJ17] J.-H. Hours and C. N. Jones. “An Alternating Trust Region Algorithm for Distributed Linearly Constrained Nonlinear Programs, Application to the Optimal Power Flow Problem”. In: *Journal of Optimization Theory and Applications* 173.3 (June 2017), pp. 844–877. DOI: 10.1007/s10957-015-0853-2.

[Hou+17] B. Houska, D. Kouzoupis, Y. Jiang, and M. Diehl. “Convex Optimization with ALADIN”. In: *Optimization Online preprint,* http://www.optimization-online.org/DB_HTML/2017/01/5827.html (2017).

[HSS17] J. Hübner, M. Schmidt, and M. C. Steinbach. “A Distributed Interior-Point KKT Solver for Multistage Stochastic Optimization”. In: *INFORMS Journal on Computing* 29.4 (Aug. 2017), pp. 612–630. DOI: 10.1287/ijoc.2017.0748.

[Köh+17] J. Köhler, M. A. Müller, N. Li, and F. Allgöwer. “Real Time Economic Dispatch for Power Networks: A Distributed Economic Model Predictive Control Approach”. In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. Dec. 2017, pp. 6340–6345. DOI: 10.1109/CDC.2017.8264615.

[Liu+17] Y. Liu, J.-H. Hours, G. Stathopoulos, and C. N. Jones. “Real-Time Distributed Algorithms for Nonconvex Optimal Power Flow”. In: *2017 American Control Conference (ACC)*. May 2017, pp. 3380–3385. DOI: 10.23919/ACC.2017.7963469.

[MO17] A. Makhdoumi and A. Ozdaglar. “Convergence Rate of Distributed ADMM Over Networks”. In: *IEEE Transactions on Automatic Control* 62.10 (Oct. 2017), pp. 5082–5095. DOI: 10.1109/TAC.2017.2677879.

[Meh+17] M. W. Mehrez, T. Sprodowski, K. Worthmann, G. Mann, R. G. Gosine, J. K. Sagawa, and J. Pannek. “Occupancy Grid Based Distributed MPC for Mobile Robots”. In: *International Conference on Intelligent Robots and Systems (IROS)*. IEEE/RSJ, 2017, pp. 4842–4847.

[MLR17] A. Mokhtari, Q. Ling, and A. Ribeiro. “Network Newton Distributed Optimization Methods”. In: *IEEE Transactions on Signal Processing* 65.1 (Jan. 2017), pp. 146–161. DOI: 10.1109/TSP.2016.2617829.

[Mol+17] D. K. Molzahn, F. Dörfler, H. Sandberg, S. H. Low, S. Chakrabarti, R. Baldick, and J. Lavaei. "A Survey of Distributed Optimization and Control Algorithms for Electric Power Systems". In: *IEEE Trans Smart Grid* 8.6 (Nov. 2017), pp. 2941–2962. DOI: 10.1109/TSG.2017.2720471.

[M+17] T. Mühlpfordt, T. Faulwasser, L. Roald, and V. Hagenmeyer. "Solving optimal power flow with non-Gaussian uncertainties via polynomial chaos expansion". In: *Proc. 56nd IEEE Conf. Decision and Control*. 2017, pp. 4490–4496. DOI: 10.1109/CDC.2017.8264321.

[MA17] M. A. Müller and F. Allgöwer. "Economic and Distributed Model Predictive Control: Recent Developments in Optimization-Based Control". In: *SICE Journal of Control, Measurement, and System Integration* 10.2 (2017), pp. 39–52. DOI: 10.9746/jcmsi.10.39.

[NOS17] A. Nedić, A. Olshevsky, and W. Shi. "Achieving Geometric Convergence for Distributed Optimization Over Time-Varying Graphs". In: *SIAM Journal on Optimization* 27.4 (Jan. 2017), pp. 2597–2633. DOI: 10.1137/16M1084316.

[Ned+17] A. Nedić, A. Olshevsky, W. Shi, and C. A. Uribe. "Geometrically Convergent Distributed Optimization with Uncoordinated Step-Sizes". In: *2017 American Control Conference (ACC)*. May 2017, pp. 3950–3955. DOI: 10.23919/ACC.2017.7963560.

[PL17] Q. Peng and S. H. Low. "Distributed Optimal Power Flow Algorithm for Radial Networks, I: Balanced Single Phase Case". In: *IEEE Trans Smart Grid* (2017), pp. 1–11. DOI: 10.1109/TSG.2016.2546305.

[RMD17] J. B. Rawlings, D. Q. Mayne, and M. Diehl. *Model Predictive Control: Theory, Computation, and Design, 2nd Edition*. Nob Hill Publishing, 2017.

[RCG17] R. Rostami, G. Costantini, and D. Görges. "ADMM-Based Distributed Model Predictive Control: Primal and Dual Approaches". In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. Dec. 2017, pp. 6598–6603. DOI: 10.1109/CDC.2017.8264654.

[Sch17] A. J. Schwab. *Elektroenergiesysteme: Erzeugung, Übertragung Und Verteilung Elektrischer Energie*. Springer-Verlag, 2017.

[Ste17] O. Stein. *Grundzüge Der Globalen Optimierung*. Springer-Verlag, 2017.

[TT17] T. Tatarenko and B. Touri. "Non-Convex Distributed Optimization". In: *IEEE Trans Autom Control* 62.8 (Aug. 2017), pp. 3744–3757. DOI: 10.1109/TAC.2017.2648041.

[Ver+17] F. Verbosio, A. D. Coninck, D. Kourounis, and O. Schenk. “Enhancing the Scalability of Selected Inversion Factorization Algorithms in Genomic Prediction”. In: *Journal of Computational Science* 22 (2017), pp. 99–108.

[BZ16] S. Bolognani and S. Zampieri. “On the Existence and Linear Approximation of the Power Flow Solution in Power Distribution Networks”. In: *IEEE Transactions on Power Systems* 31.1 (Jan. 2016), pp. 163–172. DOI: 10.1109/TPWRS.2015.2395452.

[CSL16] Y. Cao, A. Seth, and C. D. Laird. “An Augmented Lagrangian Interior-Point Approach for Large-Scale NLP Problems on Graphics Processing Units”. In: *Computers & Chemical Engineering* 85 (Feb. 2016), pp. 76–83. DOI: 10.1016/j.compchemeng.2015.10.010.

[Cap16] F. Capitanescu. “Critical Review of Recent Advances and Further Developments Needed in AC Optimal Power Flow”. In: *Electric Power Systems Research* 136 (July 2016), pp. 57–68. DOI: 10.1016/j.epsr.2016.02.008.

[Cha+16a] T.-H. Chang, M. Hong, W.-C. Liao, and X. Wang. “Asynchronous Distributed ADMM for Large-Scale Optimization—Part I: Algorithm and Convergence Analysis”. In: *IEEE Transactions on Signal Processing* 64.12 (June 2016), pp. 3118–3130. DOI: 10.1109/TSP.2016.2537271.

[Cha+16b] T.-H. Chang, W.-C. Liao, M. Hong, and X. Wang. “Asynchronous Distributed ADMM for Large-Scale Optimization—Part II: Linear Convergence Analysis and Numerical Performance”. In: *IEEE Transactions on Signal Processing* 64.12 (June 2016), pp. 3131–3144. DOI: 10.1109/TSP.2016.2537261.

[DY16] W. Deng and W. Yin. “On the Global and Linear Convergence of the Generalized Alternating Direction Method of Multipliers”. In: *Journal of Scientific Computing* 66.3 (2016), pp. 889–916.

[DLS16] P. Di Lorenzo and G. Scutari. “NEXT: In-Network Nonconvex Optimization”. In: *IEEE Transactions on Signal and Information Processing over Networks* 2.2 (June 2016), pp. 120–136. DOI: 10.1109/TSIPN.2016.2524588.

[Die16] M. Diehl. “Lecture Notes on Numerical Optimization”. 2016. URL: https://www.syscop.de/files/2015ws/numopt/numopt_0.pdf (visited on 04/08/2020).

[FR16] S. Frank and S. Rebennack. “An Introduction to Optimal Power Flow: Theory, Formulation, and Examples”. In: *IIE Transactions* 48.12 (Dec. 2016), pp. 1172–1197. DOI: 10.1080/0740817X.2016.1189626.

[GB16] P. Giselsson and S. Boyd. “Linear Convergence and Metric Selection for Douglas-Rachford Splitting and ADMM”. In: *IEEE Transactions on Automatic Control* 62.2 (2016), pp. 532–544.

[HLR16] M. Hong, Z.-Q. Luo, and M. Razaviyayn. “Convergence Analysis of Alternating Direction Method of Multipliers for a Family of Nonconvex Problems”. In: *SIAM Journal on Optimization* 26.1 (Jan. 2016), pp. 337–364. DOI: 10.1137/140990309.

[HJ16] J.-H. Hours and C. N. Jones. “A Parametric Nonconvex Decomposition Algorithm for Real-Time and Distributed NMPC”. In: *IEEE Transactions on Automatic Control* 61.2 (Feb. 2016), pp. 287–302. DOI: 10.1109/TAC.2015.2426231.

[HFD16] B. Houska, J. Frasch, and M. Diehl. “An Augmented Lagrangian Based Algorithm for Distributed Nonconvex Optimization”. In: *SIAM Journal on Optimization* 26.2 (2016), pp. 1101–1127.

[KDS16] B. Kocuk, S. S. Dey, and X. A. Sun. “Inexactness of SDP Relaxation and Valid Inequalities for Optimal Power Flow”. In: *IEEE Transactions on Power Systems* 31.1 (Jan. 2016), pp. 642–651. DOI: 10.1109/TPWRS.2015.2402640.

[Kou+16] D. Kouzoupis, R. Quirynen, B. Houska, and M. Diehl. “A Block Based ALADIN Scheme for Highly Parallelizable Direct Optimal Control”. In: *Proc. American Control Conf. (ACC)*. July 2016, pp. 1124–1129. DOI: 10.1109/ACC.2016.7525066.

[Mok+16] A. Mokhtari, W. Shi, Q. Ling, and A. Ribeiro. “A Decentralized Second-Order Method with Exact Linear Convergence Rate for Consensus Optimization”. In: *IEEE Transactions on Signal and Information Processing over Networks* 2.4 (Dec. 2016), pp. 507–522. DOI: 10.1109/TSIPN.2016.2613678.

[NOB16] R. Narain, M. Overby, and G. E. Brown. “ADMM ⊇ Projective Dynamics: Fast Simulation of General Constitutive Models”. In: *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA ’16. Zurich, Switzerland: Eurographics Association, July 2016, pp. 21–28.

[Sta+16] G. Stathopoulos, H. Shukla, A. Szucs, Y. Pu, and C. N. Jones. “Operator Splitting Methods in Control”. In: *Foundations and Trends® in Systems and Control* 3.3 (Aug. 2016), pp. 249–362. DOI: 10.1561/2600000008.

[SSP16] Y. Sun, G. Scutari, and D. Palomar. “Distributed Nonconvex Multiagent Optimization over Time-Varying Networks”. In: *2016 50th Asilomar Conference on Signals, Systems and Computers*. Nov. 2016, pp. 788–794. DOI: 10.1109/ACSSC.2016.7869154.

[Tay+16] G. Taylor, R. Burmeister, Z. Xu, B. Singh, A. Patel, and T. Goldstein. “Training Neural Networks without Gradients: A Scalable Admm Approach”. In: *International Conference on Machine Learning*. 2016, pp. 2722–2731.

[TND16] Q. Tran-Dinh, I. Necoara, and M. Diehl. “Fast Inexact Decomposition Algorithms for Large-Scale Separable Convex Optimization”. In: *Optimization* 65.2 (Feb. 2016), pp. 325–356. DOI: 10.1080/02331934.2015.1044898.

[Var+16] D. Varagnolo, F. Zanella, A. Cenedese, G. Pillonetto, and L. Schenato. “Newton-Raphson Consensus for Distributed Convex Optimization”. In: *IEEE Transactions on Automatic Control* 61.4 (Apr. 2016), pp. 994–1009. DOI: 10.1109/TAC.2015.2449811.

[Yan+16] Y. Yang, J. Sun, H. Li, and Z. Xu. “Deep ADMM-Net for Compressive Sensing MRI”. In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett. Curran Associates, Inc., 2016, pp. 10–18.

[YLY16] K. Yuan, Q. Ling, and W. Yin. “On the Convergence of Decentralized Gradient Descent”. In: *SIAM Journal on Optimization* 26.3 (2016), pp. 1835–1854.

[Ber15a] D. P. Bertsekas. *Convex Optimization Algorithms*. Athena Scientific Belmont, 2015.

[Ber15b] D. P. Bertsekas. *Incremental Aggregated Proximal and Augmented Lagrangian Algorithms*. Report Laboratory for Information and Decision Systems 3176. Cambridge: M.I.T., Nov. 2015. arXiv: 1509.09257.

[CDZ15] N. Chatzipanagiotis, D. Dentcheva, and M. M. Zavlanos. “An Augmented Lagrangian Method for Distributed Optimization”. In: *Mathematical Programming* 152.1 (Aug. 2015), pp. 405–434. DOI: 10.1007/s10107-014-0808-7.

[Ers15] T. Erseghe. “A Distributed Approach to the OPF Problem”. In: *EURASIP Journal on Advances in Signal Processing* 2015.1 (May 2015), p. 45. DOI: 10.1186/s13634-015-0226-x.

[Fei+15] M. J. Feizollahi, M. Costley, S. Ahmed, and S. Grijalva. “Large-Scale Decentralized Unit Commitment”. In: *International Journal of Electrical Power & Energy Systems* 73 (Dec. 2015), pp. 97–106. DOI: 10.1016/j.ijepes.2015.04.009.

[FSD15] J. V. Frasch, S. Sager, and M. Diehl. “A Parallel Quadratic Programming Method for Dynamic Optimization Problems”. In: *Mathematical Programming Computation* 7.3 (Sept. 2015), pp. 289–329. DOI: 10.1007/s12532-015-0081-7.

[GB15] P. Giselsson and S. Boyd. “Metric Selection in Fast Dual Forward–Backward Splitting”. In: *Automatica* 62 (Dec. 2015), pp. 1–10. DOI: 10.1016/j.automatica.2015.09.010.

[HY15] B. He and X. Yuan. “On Non-Ergodic Convergence Rate of Douglas–Rachford Alternating Direction Method of Multipliers”. In: *Numerische Mathematik* 130.3 (2015), pp. 567–577.

[HKW15] G. Hug, S. Kar, and C. Wu. “Consensus + Innovations Approach for Distributed Multiagent Coordination in a Microgrid”. In: *IEEE Trans Smart Grid* 6.4 (July 2015), pp. 1893–1903. DOI: 10.1109/TSG.2015.2409053.

[KCD15] A. Kozma, C. Conte, and M. Diehl. “Benchmarking Large-Scale Distributed Convex Quadratic Programming Algorithms”. In: *Optimization Methods and Software* 30.1 (Jan. 2015), pp. 191–214. DOI: 10.1080/10556788.2014.911298.

[Lei+15] T. Leibfried, T. Mchedlidze, N. Meyer-Hübner, M. Nöllenburg, I. Rutter, P. Sanders, D. Wagner, and F. Wegner. “Operating Power Grids with Few Flow Control Buses”. In: *Proceedings of the 2015 ACM Sixth International Conference on Future Energy Systems*. E-Energy ’15. Bangalore, India: Association for Computing Machinery, July 2015, pp. 289–294. DOI: 10.1145/2768510.2768521.

[LW15] B. Lesieutre and D. Wu. “An Efficient Method to Locate All the Load Flow Solutions - Revisited”. In: *53rd Annual Allerton Conference on Communication, Control, and Computing*. Sept. 2015, pp. 381–388. DOI: 10.1109/ALLERTON.2015.7447029.

[LP15] G. Li and T. K. Pong. “Global Convergence of Splitting Methods for Nonconvex Composite Optimization”. In: *SIAM Journal on Optimization* 25.4 (Jan. 2015), pp. 2434–2460. DOI: 10.1137/140998135.

[LMZ15] T. Lin, S. Ma, and S. Zhang. “On the Global Linear Convergence of the ADMM with Multiblock Variables”. In: *SIAM Journal on Optimization* 25.3 (2015), pp. 1478–1497.

[Sch15] K. Schittkowski. *NLPQLP: A Fortran Implementation of a Sequential Quadratic Programming Algorithm with Distributed and Non-Monotone Line Search - User's Guide, Version 4.2* -. Tech. rep. 2015.

[Shi+15a] W. Shi, Q. Ling, G. Wu, and W. Yin. "Extra: An Exact First-Order Algorithm for Decentralized Consensus Optimization". In: *SIAM Journal on Optimization* 25.2 (2015), pp. 944–966.

[Shi+15b] W. Shi, Q. Ling, G. Wu, and W. Yin. "A Proximal Gradient Algorithm for Decentralized Composite Optimization". In: *IEEE Transactions on Signal Processing* 63.22 (Nov. 2015), pp. 6013–6023. DOI: 10.1109/TSP.2015.2461520.

[Xu+15] J. Xu, S. Zhu, Y. C. Soh, and L. Xie. "Augmented Distributed Gradient Methods for Multi-Agent Optimization under Uncoordinated Constant Stepsizes". In: *2015 54th IEEE Conference on Decision and Control (CDC)*. Dec. 2015, pp. 2055–2060. DOI: 10.1109/CDC.2015.7402509.

[Zhu15] J. Zhu. *Optimization of Power System Operation*. Vol. 47. John Wiley & Sons, 2015.

[BCH14] D. Bienstock, M. Chertkov, and S. Harnett. "Chance-Constrained Optimal Power Flow: Risk-Aware Network Control under Uncertainty". In: *SIAM Rev* 56.3 (2014), pp. 461–495. DOI: 10.1137/130910312.

[Cai+14] X. Cai, M. Tippett, L. Xie, and J. Bao. "Fast Distributed MPC Based on Active Set Method". In: *Computers & Chemical Engineering* 71 (2014), pp. 158–170.

[CPZ14] N. Chiang, C. G. Petra, and V. M. Zavala. "Structured Nonconvex Optimization of Large-Scale Energy Systems Using PIPS-NLP". In: *2014 Power Systems Computation Conference*. Aug. 2014, pp. 1–7. DOI: 10.1109/PSCC.2014.7038374.

[Ers14a] T. Erseghe. "Distributed Optimal Power Flow Using ADMM". In: *IEEE Trans Power Syst* 29.5 (Sept. 2014), pp. 2370–2380. DOI: 10.1109/TPWRS.2014.2306495.

[Ers14b] T. Erseghe. "Distributed Optimal Power Flow Using ADMM". In: *IEEE Transactions on Power Systems* 29.5 (Sept. 2014), pp. 2370–2380. DOI: 10.1109/TPWRS.2014.2306495.

[Gol+14] T. Goldstein, B. O'Donoghue, S. Setzer, and R. G. Baraniuk. "Fast Alternating Direction Optimization Methods". In: *SIAM Journal on Imaging Sciences* 7 (2014), pp. 1588–1623.

[HJ14] J.-H. Hours and C. N. Jones. “An Augmented Lagrangian Coordination-Decomposition Algorithm for Solving Distributed Non-Convex Programs”. In: *2014 American Control Conference*. June 2014, pp. 4312–4317. DOI: 10.1109/ACC.2014.6858863.

[JXM14] D. Jakovetic, J. Xavier, and J. M. F. Moura. “Fast Distributed Gradient Methods”. In: *IEEE Transactions on Automatic Control* 59.5 (May 2014), pp. 1131–1146. DOI: 10.1109/TAC.2014.2298712.

[Kan+14] J. Kang, Y. Cao, D. P. Word, and C. D. Laird. “An Interior-Point Method for Efficient Solution of Block-Structured NLP Problems Using an Implicit Schur-Complement Decomposition”. In: *Computers & Chemical Engineering* 71 (Dec. 2014), pp. 563–573. DOI: 10.1016/j.compchemeng.2014.09.013.

[Low14a] S. H. Low. “Convex Relaxation of Optimal Power Flow —Part I: Formulations and Equivalence”. In: *IEEE Transactions on Control of Network Systems* 1.1 (Mar. 2014), pp. 15–27. DOI: 10.1109/TCNS.2014.2309732.

[Low14b] S. H. Low. “Convex Relaxation of Optimal Power Flow —Part II: Exactness”. In: *IEEE Transactions on Control of Network Systems* 1.2 (June 2014), pp. 177–189. DOI: 10.1109/TCNS.2014.2323634.

[Ned14] A. Nedić. “Distributed Optimization Over Networks”. In: *Multi-Agent Optimization*. Ed. by F. Facchinei and J.-S. Pang. Springer International Publishing, 2014, pp. 1–84.

[Ned+14] A. Nedić, J.-S. Pang, G. Scutari, and Y. Sun. *Multi-Agent Optimization: Cetraro, Italy 2014*. Ed. by F. Facchinei and J.-S. Pang. Vol. 2224. Lecture Notes in Mathematics. Cham: Springer International Publishing, 2014. DOI: 10.1007/978-3-319-97142-1.

[Och+14] P. Ochs, Y. Chen, T. Brox, and T. Pock. “iPiano: Inertial Proximal Algorithm for Nonconvex Optimization”. In: *SIAM Journal on Imaging Sciences* 7.2 (Jan. 2014), pp. 1388–1419. DOI: 10.1137/130942954.

[PHA14] S. K. Pakazad, A. Hansson, and M. S. Andersen. “Distributed Interior-Point Method for Loosely Coupled Problems”. In: *IFAC Proceedings Volumes*. 19th IFAC World Congress 47.3 (Jan. 2014), pp. 9587–9592. DOI: 10.3182/20140824-6-ZA-1003.01647.

[PB14] N. Parikh and S. Boyd. “Proximal Algorithms”. In: *Foundations and Trends® in Optimization* 1.3 (2014), pp. 127–239.

[PSB14] P. Patrinos, L. Stella, and A. Bemporad. “Douglas-Rachford Splitting: Complexity Estimates and Accelerated Variants”. In: *53rd IEEE Conference on Decision and Control*. Dec. 2014, pp. 4234–4239. DOI: 10.1109/CDC.2014.7040049.

[Scu+14] G. Scutari, F. Facchinei, P. Song, D. P. Palomar, and J.-S. Pang. “Decomposition by Partial Linearization: Parallel Optimization of Multi-Agent Systems”. In: *IEEE Transactions on Signal Processing* 62.3 (Feb. 2014), pp. 641–656. DOI: 10.1109/TSP.2013.2293126.

[ST14] R. Shefi and M. Teboulle. “Rate of Convergence Analysis of Decomposition Methods Based on the Proximal Method of Multipliers for Convex Minimization”. In: *SIAM Journal on Optimization* 24.1 (Jan. 2014), pp. 269–297. DOI: 10.1137/130910774.

[Shi+14] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin. “On the Linear Convergence of the ADMM in Decentralized Consensus Optimization”. In: *IEEE Transactions on Signal Processing* 62.7 (2014), pp. 1750–1761.

[ŠBC14] P. Šulc, S. Backhaus, and M. Chertkov. “Optimal Distributed Control of Reactive Power Via the Alternating Direction Method of Multipliers”. In: *IEEE Transactions on Energy Conversion* 29.4 (Dec. 2014), pp. 968–977. DOI: 10.1109/TEC.2014.2363196.

[Wor+14] D. P. Word, J. Kang, J. Akesson, and C. D. Laird. “Efficient Parallel Solution of Large-Scale Nonlinear Dynamic Optimization Problems”. In: *Computational Optimization and Applications* 59.3 (Dec. 2014), pp. 667–688. DOI: 10.1007/s10589-014-9651-2.

[ZK14] R. Zhang and J. T. Kwok. “Asynchronous Distributed ADMM for Consensus Optimization”. In: *Proceedings of the 31st International Conference on Machine Learning - Volume 32*. ICML’14. Beijing, China: JMLR.org, June 2014, pp. II–1701–II–1709.

[ABS13] H. Attouch, J. Bolte, and B. F. Svaiter. “Convergence of Descent Methods for Semi-Algebraic and Tame Problems: Proximal Algorithms, Forward–Backward Splitting, and Regularized Gauss–Seidel Methods”. In: *Mathematical Programming* 137.1 (Feb. 2013), pp. 91–129. DOI: 10.1007/s10107-011-0484-9.

[BSS13] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms*. John Wiley & Sons, 2013.

[Buk+13] W. A. Bukhsh, A. Grothey, K. I. M. McKinnon, and P. A. Trodden. “Local Solutions of the Optimal Power Flow Problem”. In: *IEEE Transactions on Power Systems* 28.4 (Nov. 2013), pp. 4780–4788. DOI: 10.1109/TPWRS.2013.2274577.

[DZG13a] E. Dall'Anese, H. Zhu, and G. B. Giannakis. "Distributed Optimal Power Flow for Smart Microgrids". In: *IEEE Trans Smart Grid* 4.3 (Sept. 2013), pp. 1464–1475. DOI: 10.1109/TSG.2013.2248175.

[DZG13b] E. Dall'Anese, H. Zhu, and G. B. Giannakis. "Distributed Optimal Power Flow for Smart Microgrids". In: *IEEE Transactions on Smart Grid* 4.3 (Sept. 2013), pp. 1464–1475. DOI: 10.1109/TSG.2013.2248175.

[Din+13] Q. T. Dinh, I. Necoara, C. Savorgnan, and M. Diehl. "An Inexact Perturbed Path-Following Method for Lagrangian Decomposition in Large-Scale Separable Convex Optimization". In: *SIAM Journal on Optimization* 23.1 (Jan. 2013), pp. 95–125. DOI: 10.1137/11085311X.

[LC13] S. Lin and J. Chen. "Distributed Optimal Power Flow for Smart Grid Transmission System with Renewable Energy Sources". In: *Energy* 56 (2013), pp. 184–192.

[Mol+13] D. K. Molzahn, J. T. Holzer, B. C. Lesieutre, and C. L. DeMarco. "Implementation of a Large-Scale Optimal Power Flow Solver Based on Semidefinite Programming". In: *IEEE Trans Power Syst* 28.4 (Nov. 2013), pp. 3987–3998. DOI: 10.1109/TPWRS.2013.2258044.

[Nes13] Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Vol. 87. Springer Science & Business Media, 2013.

[RHL13] M. Razaviyayn, M. Hong, and Z.-Q. Luo. "A Unified Convergence Analysis of Block Successive Minimization Methods for Nonsmooth Optimization". In: *SIAM Journal on Optimization* 23.2 (Jan. 2013), pp. 1126–1153. DOI: 10.1137/120891009.

[SPG13] A. X. Sun, D. T. Phan, and S. Ghosh. "Fully Decentralized AC Optimal Power Flow Algorithms". In: *2013 IEEE Power Energy Society General Meeting*. July 2013, pp. 1–5. DOI: 10.1109/PESMG.2013.6672864.

[TDSD13] Q. Tran Dinh, C. Savorgnan, and M. Diehl. "Combining Lagrangian Decomposition and Excessive Gap Smoothing Technique for Solving Large-Scale Separable Convex Optimization Problems". In: *Computational Optimization and Applications* 55.1 (May 2013), pp. 75–111. DOI: 10.1007/s10589-012-9515-6.

[WO13] E. Wei and A. Ozdaglar. "On the O(1/k) Convergence of Asynchronous Distributed Alternating Direction Method of Multipliers". In: *2013 IEEE Global Conference on Signal and Information Processing*. Dec. 2013, pp. 551–554.

[WW13] A. J. Wood and B. F. Wollenberg. *Power Generation, Operation, and Control*. John Wiley & Sons, 2013.

[XY13] Y. Xu and W. Yin. “A Block Coordinate Descent Method for Regularized Multiconvex Optimization with Applications to Nonnegative Tensor Factorization and Completion”. In: *SIAM Journal on Imaging Sciences* 6.3 (Jan. 2013), pp. 1758–1789. DOI: 10.1137/120887795.

[CO12] A. I. Chen and A. Ozdaglar. “A Fast Distributed Proximal-Gradient Method”. In: *2012 50th Annual Allerton Conference on Communication, Control, and Computing*. Oct. 2012, pp. 601–608. DOI: 10.1109/Allerton.2012.6483273.

[Cur+12] F. E. Curtis, J. Huber, O. Schenk, and A. Wächter. “A Note on the Implementation of an Interior-Point Algorithm for Nonlinear Optimization with Inexact Step Computations”. In: *Mathematical Programming* 136.1 (Dec. 2012), pp. 209–227. DOI: 10.1007/s10107-012-0557-4.

[DAW12] J. C. Duchi, A. Agarwal, and M. J. Wainwright. “Dual Averaging for Distributed Optimization: Convergence Analysis and Network Scaling”. In: *IEEE Transactions on Automatic Control* 57.3 (Mar. 2012), pp. 592–606. DOI: 10.1109/TAC.2011.2161027.

[FSR12] S. Frank, I. Steponavice, and S. Rebennack. “Optimal Power Flow: A Bibliographic Survey I”. In: *Energy Systems* 3.3 (Sept. 2012), pp. 221–258. DOI: 10.1007/s12667-012-0056-y.

[GM12] D. Goldfarb and S. Ma. “Fast Multiple-Splitting Algorithms for Convex Optimization”. In: *SIAM Journal on Optimization* 22.2 (Jan. 2012), pp. 533–556. DOI: 10.1137/090780705.

[HY12] B. He and X. Yuan. “On the O(1/n) Convergence Rate of the Douglas–Rachford Alternating Direction Method”. In: *SIAM Journal on Numerical Analysis* 50.2 (Jan. 2012), pp. 700–709. DOI: 10.1137/110836936.

[LL12] J. Lavaei and S. H. Low. “Zero Duality Gap in Optimal Power Flow Problem”. In: *IEEE Trans Power Syst* 27.1 (Feb. 2012), pp. 92–107. DOI: 10.1109/TPWRS.2011.2160974.

[SNW12] S. Sra, S. Nowozin, and S. J. Wright, eds. *Optimization for Machine Learning*. Neural Information Processing Series. Cambridge, Mass: MIT Press, 2012.

[WO12] E. Wei and A. Ozdaglar. “Distributed Alternating Direction Method of Multipliers”. In: *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. Dec. 2012, pp. 5445–5450. DOI: 10.1109/CDC.2012.6425904.

[ZM12] M. Zhu and S. Martinez. “On Distributed Convex Optimization Under Inequality and Equality Constraints”. In: *IEEE Transactions on Automatic Control* 57.1 (Jan. 2012), pp. 151–164. DOI: 10.1109/TAC.2011.2167817.

[BC11] H. H. Bauschke and P. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Vol. 408. Springer, 2011. DOI: 10.1007/978-1-4419-9467-7.

[Boy+11] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. “Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers”. In: *Foundations and Trends® in Machine Learning* 3.1 (2011), pp. 1–122.

[Cap+11] F. Capitanescu, J. L. Martinez Ramos, P. Panciatici, D. Kirschen, A. Marano Marcolini, L. Platbrood, and L. Wehenkel. “State-of-the-Art, Challenges, and Future Trends in Security Constrained Optimal Power Flow”. In: *Electric Power Systems Research* 81.8 (Aug. 2011), pp. 1731–1741. DOI: 10.1016/j.epsr.2011.04.003.

[CL11] C.-C. Chang and C.-J. Lin. “LIBSVM: A Library for Support Vector Machines”. In: *ACM Transactions on Intelligent Systems and Technology* 2.3 (May 2011), 27:1–27:27. DOI: 10.1145/1961189.1961199.

[CLM11] P. Christofides, J. Liu, and D. Munoz de la Pena. *Networked and Distributed Predictive Control: Methods and Nonlinear Process Network Applications*. Springer Science & Business Media, 2011.

[CP11] P. L. Combettes and J.-C. Pesquet. “Proximal Splitting Methods in Signal Processing”. In: *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*. Ed. by H. H. Bauschke, R. S. Burachik, P. L. Combettes, V. Elser, D. R. Luke, and H. Wolkowicz. Springer Optimization and Its Applications. New York, NY: Springer, 2011, pp. 185–212. DOI: 10.1007/978-1-4419-9569-8_10.

[Glo11] J. D. Glover. *Power System Analysis and Design*. Fifth. Cengage Learning, 2011.

[HM11] A. Hamdi and S. K. Mishra. “Decomposition Methods Based on Augmented Lagrangians: A Survey”. In: *Topics in Nonconvex Optimization: Theory and Applications*. Ed. by S. K. Mishra. Springer Optimization and Its Applications. New York, NY: Springer, 2011, pp. 175–203. DOI: 10.1007/978-1-4419-9640-4_11.

[Lub+11] M. Lubin, C. G. Petra, M. Anitescu, and V. Zavala. “Scalable Stochastic Optimization of Complex Energy Systems”. In: *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. Nov. 2011, pp. 1–10. DOI: 10.1145/2063384.2063470.

[NND11] I. Necoara, V. Nedelcu, and I. Dumitrache. “Parallel and Distributed Optimization Methods for Estimation and Control in Networks”. In: *Journal of Process Control*. Special Issue on Hierarchical and Distributed Model Predictive Control 21.5 (June 2011), pp. 756–766. DOI: 10.1016/j.jprocont.2010.12.010.

[ND11] I. N. V. Nedelcu and I. Dumitrache. “Parallel and Distributed Optimization Methods for Estimation and Control in Networks”. In: *Journal of Process Control* 21.5 (2011), pp. 756–766.

[Ned11] A. Nedic. “Asynchronous Broadcast-Based Convex Optimization Over a Network”. In: *IEEE Transactions on Automatic Control* 56.6 (June 2011), pp. 1337–1351. DOI: 10.1109/TAC.2010.2079650.

[SWR11] B. T. Stewart, S. J. Wright, and J. B. Rawlings. “Cooperative Distributed Model Predictive Control for Nonlinear Systems”. In: *Journal of Process Control* 21.5 (2011), pp. 698–704.

[TTM11] H. Terelius, U. Topcu, and R. M. Murray. “Decentralized Multi-Agent Optimization via Dual Decomposition”. In: *IFAC Proceedings Volumes*. 18th IFAC World Congress 44.1 (Jan. 2011), pp. 11245–11251. DOI: 10.3182/20110828-6-IT-1002.01959.

[Yan+11] W. Yan, L. Wen, W. Li, C. Y. Chung, and K. P. Wong. “Decomposition–Coordination Interior Point Method and Its Application to Multi-Area Optimal Reactive Power Flow”. In: *International Journal of Electrical Power & Energy Systems* 33.1 (Jan. 2011), pp. 55–60. DOI: 10.1016/j.ijepes.2010.08.004.

[ABF10] M. V. Afonso, J. M. Bioucas-Dias, and M. A. T. Figueiredo. “Fast Image Recovery Using Variable Splitting and Constrained Optimization”. In: *IEEE Transactions on Image Processing* 19.9 (Sept. 2010), pp. 2345–2356. DOI: 10.1109/TIP.2010.2047910.

[DAW10] J. C. Duchi, A. Agarwal, and M. J. Wainwright. “Distributed Dual Averaging In Networks”. In: *Advances in Neural Information Processing Systems 23*. Ed. by J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta. Curran Associates, Inc., 2010, pp. 550–558.

[FCG10] P. A. Forero, A. Cano, and G. B. Giannakis. “Consensus-Based Distributed Support Vector Machines”. In: *Journal of Machine Learning Research* (2010), p. 45.

[LB10] F. Li and R. Bo. “Small Test Systems for Power System Economic Studies”. In: *IEEE PES General Meeting*. July 2010, pp. 1–4.

[MBG10] G. Mateos, J. A. Bazerque, and G. B. Giannakis. “Distributed Sparse Linear Regression”. In: *IEEE Transactions on Signal Processing* 58.10 (Oct. 2010), pp. 5262–5276. DOI: 10.1109/TSP.2010.2055862.

[NOP10] A. Nedic, A. Ozdaglar, and P. A. Parrilo. “Constrained Consensus and Optimization in Multi-Agent Networks”. In: *IEEE Transactions on Automatic Control* 55.4 (Apr. 2010), pp. 922–938. DOI: 10.1109/TAC.2010.2041686.

[SMG10] K. Scheinberg, S. Ma, and D. Goldfarb. “Sparse Inverse Covariance Selection via Alternating Linearization Methods”. In: *Advances in Neural Information Processing Systems 23*. 2010, p. 9.

[SRNV10] S. Sundhar Ram, A. Nedić, and V. V. Veeravalli. “Distributed Stochastic Subgradient Projection Algorithms for Convex Optimization”. In: *Journal of Optimization Theory and Applications* 147.3 (Dec. 2010), pp. 516–545. DOI: 10.1007/s10957-010-9737-7.

[ZA10] V. M. Zavala and M. Anitescu. “Real-Time Nonlinear Optimization as a Generalized Equation”. In: *SIAM Journal on Control and Optimization* 48.8 (Jan. 2010), pp. 5444–5467. DOI: 10.1137/090762634.

[ZM10] M. Zhu and S. Martínez. “Discrete-Time Dynamic Average Consensus”. In: *Automatica* 46.2 (Feb. 2010), pp. 322–329. DOI: 10.1016/j.automatica.2009.10.021.

[BT09] A. Beck and M. Teboulle. “A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems”. In: *SIAM Journal on Imaging Sciences* 2.1 (Jan. 2009), pp. 183–202. DOI: 10.1137/080716542.

[GO09] T. Goldstein and S. Osher. “The Split Bregman Method for L1-Regularized Problems”. In: *SIAM J. Imaging Sciences* 2 (Jan. 2009), pp. 323–343. DOI: 10.1137/080725891.

[GG09] J. Gondzio and A. Grothey. “Exploiting Structure in Parallel Implementation of Interior Point Methods for Optimization”. In: *Computational Management Science* 6.2 (May 2009), pp. 135–160. DOI: 10.1007/s10287-008-0090-3.

[HA09] G. Hug-Glanzmann and G. Andersson. “Decentralized Optimal Power Flow Control for Overlapping Areas in Power Systems”. In: *IEEE Transactions on Power Systems* 24.1 (Feb. 2009), pp. 327–336. DOI: 10.1109/TPWRS.2008.2006998.

[JOZ09] A. Jadbabaie, A. Ozdaglar, and M. Zargham. “A Distributed Newton Method for Network Optimization”. In: *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) Held Jointly with 2009 28th Chinese Control Conference*. Dec. 2009, pp. 2736–2741. DOI: 10.1109/CDC.2009.5400289.

[NS09] I. Necoara and J. A. K. Suykens. “Interior-Point Lagrangian Decomposition Method for Separable Convex Optimization”. In: *Journal of Optimization Theory and Applications* 143.3 (May 2009), p. 567. DOI: 10.1007/s10957-009-9566-8.

[NO09] A. Nedić and A. Ozdaglar. “Cooperative Distributed Multi-Agent Optimization”. In: *Convex Optimization in Signal Processing and Communications*. Ed. by D. P. Palomar and Y. C. Eldar. Cambridge University Press, 2009, pp. 340–386. DOI: 10.1017/CBO9780511804458.011.

[Nes09] Y. Nesterov. “Primal-Dual Subgradient Methods for Convex Problems”. In: *Mathematical Programming* 120.1 (Aug. 2009), pp. 221–259. DOI: 10.1007/s10107-007-0149-x.

[OT09] A. Olshevsky and J. N. Tsitsiklis. “Convergence Speed in Distributed Consensus and Averaging”. In: *SIAM Journal on Control and Optimization* 48.1 (Jan. 2009), pp. 33–55. DOI: 10.1137/060678324.

[Sol09] M. V. Solodov. “Global Convergence of an SQP Method without Boundedness Assumptions on Any of the Iterative Sequences”. In: *Mathematical Programming* 118.1 (Apr. 2009), pp. 1–12. DOI: 10.1007/s10107-007-0180-y.

[ZGC09] H. Zhu, G. B. Giannakis, and A. Cano. “Distributed In-Network Channel Decoding”. In: *IEEE Transactions on Signal Processing* 57.10 (Oct. 2009), pp. 3970–3983. DOI: 10.1109/TSP.2009.2023936.

[Bai+08] X. Bai, H. Wei, K. Fujisawa, and Y. Wang. “Semidefinite Programming for Optimal Power Flow Problems”. In: *International Journal of Electrical Power & Energy Systems* 30.6 (July 2008), pp. 383–392. DOI: 10.1016/j.ijepes.2007.12.003.

[NS08] I. Necoara and J. A. K. Suykens. “Application of a Smoothing Technique to Decomposition in Convex Optimization”. In: *IEEE Transactions on Automatic Control* 53.11 (Dec. 2008), pp. 2674–2679. DOI: 10.1109/TAC.2008.2007159.

[SRG08] I. D. Schizas, A. Ribeiro, and G. B. Giannakis. “Consensus in Ad Hoc WSNs With Noisy Links—Part I: Distributed Estimation of Deterministic Signals”. In: *IEEE Transactions on Signal Processing* 56.1 (Jan. 2008), pp. 350–364. DOI: 10.1109/TSP.2007.906734.

[ZLB08] V. M. Zavala, C. D. Laird, and L. T. Biegler. “Interior-Point Decomposition Approaches for Parallel Solution of Large-Scale Nonlinear Parameter Estimation Problems”. In: *Chemical Engineering Science*. Model-Based Experimental Analysis 63.19 (Oct. 2008), pp. 4834–4845. DOI: 10.1016/j.ces.2007.05.022.

[AKA07] M. Arnold, S. Knopfli, and G. Andersson. “Improvement of OPF Decomposition Methods Applied to Multi-Area Power Systems”. In: *2007 IEEE Lausanne Power Tech*. July 2007, pp. 1308–1313. DOI: 10.1109/PCT.2007.4538505.

[BT07] D. P. Bertsekas and J. N. Tsitsiklis. “Comments on “Coordination of Groups of Mobile Autonomous Agents Using Nearest Neighbor Rules””. In: *IEEE Transactions on Automatic Control* 52.5 (May 2007), pp. 968–969. DOI: 10.1109/TAC.2007.895885.

[Chi+07] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle. “Layering as Optimization Decomposition: A Mathematical Theory of Network Architectures”. In: *Proceedings of the IEEE* 95.1 (Jan. 2007), pp. 255–312. DOI: 10.1109/JPROC.2006.887322.

[CMS07] B. Colson, P. Marcotte, and G. Savard. “An Overview of Bilevel Optimization”. In: *Annals of Operations Research* 153.1 (Sept. 2007), pp. 235–256. DOI: 10.1007/s10479-007-0176-2.

[CP07] P. L. Combettes and J.-C. Pesquet. “A Douglas–Rachford Splitting Approach to Nonsmooth Convex Variational Signal Recovery”. In: *IEEE Journal of Selected Topics in Signal Processing* 1.4 (Dec. 2007), pp. 564–574. DOI: 10.1109/JSTSP.2007.910264.

[GG07] J. Gondzio and A. Grothey. “Parallel Interior-Point Solver for Structured Quadratic Programs: Application to Financial Planning Problems”. In: *Annals of Operations Research* 152.1 (July 2007), pp. 319–339. DOI: 10.1007/s10479-006-0139-z.

[OFM07] R. Olfati-Saber, J. A. Fax, and R. M. Murray. “Consensus and Cooperation in Networked Multi-Agent Systems”. In: *Proceedings of the IEEE* 95.1 (Jan. 2007), pp. 215–233. DOI: 10.1109/JPROC.2006.887293.

[BB06] P. Biskas and A. Bakirtzis. “Decentralised OPF of Large Multiarea Power Systems”. In: *IEE Proceedings - Generation, Transmission and Distribution* 153.1 (Jan. 2006), pp. 99–105. DOI: 10.1049/ip-gtd:20045250.

[Con+06] A. J. Conejo, E. Castillo, R. Minguez, and R. Garcia-Bertrand. *Decomposition Techniques in Mathematical Programming: Engineering and Science Applications*. Springer Science & Business Media, Berlin/Heidelberg, 2006.

[NW06] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Science & Business Media, New York, 2006.

[Sni06] M. Sniedovich. “Dijkstra’s Algorithm Revisited: The Dynamic Programming Connexion”. In: *Control and Cybernetics* 35 (Jan. 2006).

[WB06] A. Wächter and L. T. Biegler. “On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming”. In: *Mathematical Programming* 106.1 (Mar. 2006), pp. 25–57. DOI: 10.1007/s10107-004-0559-y.

[Blo+05] V. Blondel, J. Hendrickx, A. Olshevsky, and J. Tsitsiklis. “Convergence in Multiagent Coordination, Consensus, and Flocking”. In: *Proceedings of the 44th IEEE Conference on Decision and Control*. Dec. 2005, pp. 2996–3000. DOI: 10.1109/CDC.2005.1582620.

[Boy+05] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. “Gossip Algorithms: Design, Analysis and Applications”. In: *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.* Vol. 3. Mar. 2005, 1653–1664 vol. 3. DOI: 10.1109/INFCOM.2005.1498447.

[CCD05] H. Chen, J. Chen, and X. Duan. “Multi-Stage Dynamic Optimal Power Flow in Wind Power Integrated System”. In: *2005 IEEE/PES Transmission Distribution Conference Exposition: Asia and Pacific*. Aug. 2005, pp. 1–5. DOI: 10.1109/TDC.2005.1546979.

[Nes05a] Y. Nesterov. “Excessive Gap Technique in Nonsmooth Convex Minimization”. In: *SIAM Journal on Optimization* 16.1 (Jan. 2005), pp. 235–249. DOI: 10.1137/S1052623403422285.

[Nes05b] Y. Nesterov. “Smooth Minimization of Non-Smooth Functions”. In: *Mathematical Programming* 103.1 (May 2005), pp. 127–152. DOI: 10.1007/s10107-004-0552-5.

[Boy+04] S. Boyd, S. Boyd, L. Vandenberghe, and C. U. Press. *Convex Optimization*. Berichte Über Verteilte Messysteme. Cambridge University Press, 2004.

[GGT04] G. Giorgi, A. Guerraggio, and J. Thierfelder. *Mathematics of Optimization: Smooth and Nonsmooth Case*. Elsevier, 2004.

[BB03] A. Bakirtzis and P. Biskas. "A Decentralized Solution to the DC-OPF of Interconnected Power Systems". In: *IEEE Transactions on Power Systems* 18.3 (Aug. 2003), pp. 1007–1013. DOI: 10.1109/TPWRS.2003.814853.

[Don+03] J. Dongarra, I. Foster, G. Fox, W. Gropp, K. Kennedy, L. Torczon, and A. White. *Sourcebook of Parallel Computing*. Vol. 3003. Morgan Kaufmann Publishers, San Francisco, 2003.

[JLM03] A. Jadbabaie, J. Lin, and A. Morse. "Coordination of Groups of Mobile Autonomous Agents Using Nearest Neighbor Rules". In: *IEEE Transactions on Automatic Control* 48.6 (June 2003), pp. 988–1001. DOI: 10.1109/TAC.2003.812781.

[Lei+03] D. B. Leineweber, I. Bauer, H. G. Bock, and J. P. Schlöder. "An Efficient Multiple Shooting Based Reduced SQP Strategy for Large-Scale Dynamic Process Optimization. Part 1: Theoretical Aspects". In: *Computers & Chemical Engineering* 27.2 (Feb. 2003), pp. 157–166. DOI: 10.1016/S0098-1354(02)00158-8.

[NPC03] F. J. Nogales, F. J. Prieto, and A. J. Conejo. "A Decomposition Methodology Applied to the Multi-Area Optimal Power Flow Problem". In: *Annals of Operations Research* 120.1 (Apr. 2003), pp. 99–116. DOI: 10.1023/A:1023374312364.

[Saa03] Y. Saad. *Iterative Methods for Sparse Linear Systems, 2nd Edition*. SIAM, Philadelphia, 2003.

[CNP02] A. J. Conejo, F. J. Nogales, and F. J. Prieto. "A Decomposition Procedure Based on Approximate Newton Directions". In: *Mathematical Programming* 93.3 (Dec. 2002), pp. 495–515. DOI: 10.1007/s10107-002-0304-3.

[HPK02] D. Hur, J.-K. Park, and B. Kim. "Evaluation of Convergence Rate in the Auxiliary Problem Principle for Distributed Optimal Power Flow". In: *Transmission and Distribution IEE Proceedings - Generation* 149.5 (Sept. 2002), pp. 525–532. DOI: 10.1049/ip-gtd:20020463.

[BK01] B. Beckermann and A. B. J. Kuijlaars. "Superlinear Convergence of Conjugate Gradients". In: *SIAM Journal on Numerical Analysis* 39.1 (Jan. 2001), pp. 300–329. DOI: 10.1137/S0036142999363188.

[BMN01] A. Ben-Tal, T. Margalit, and A. Nemirovski. “The Ordered Subsets Mirror Descent Optimization Method with Applications to Tomography”. In: *SIAM Journal on Optimization* 12.1 (Jan. 2001), pp. 79–108. DOI: 10.1137/S1052623499354564.

[HOS01] M. Hegland, M. Osborne, and J. Sun. “Parallel Interior Point Schemes for Solving Multistage Convex Programming”. In: *Annals of Operations Research* 108.1 (Nov. 2001), pp. 75–85. DOI: 10.1023/A:1016098709653.

[Sch+01] O. Schenk, K. Gärtner, W. Fichtner, and A. Stricker. “PARDISO: A High-Performance Serial and Parallel Sparse Linear Solver in Semiconductor Device Simulation”. In: *Future Generation Computer Systems* 18.1 (Sept. 2001), pp. 69–78. DOI: 10.1016/S0167-739X(00)00076-5.

[WSL01] X. Wang, Y. Song, and Q. Lu. “Lagrangian Decomposition Approach to Active Power Congestion Management across Interconnected Regions”. In: *Transmission and Distribution IEE Proceedings - Generation* 148.5 (Sept. 2001), pp. 497–503. DOI: 10.1049/ip-gtd:20010490.

[XS01] K. Xie and Y. Song. “Dynamic Optimal Power Flow by Interior Point Methods”. In: *Transmission and Distribution IEE Proceedings - Generation* 148.1 (Jan. 2001), pp. 76–84. DOI: 10.1049/ip-gtd:20010026.

[AC00] N. Alguacil and A. Conejo. “Multiperiod Optimal Power Flow Using Benders Decomposition”. In: *IEEE Transactions on Power Systems* 15.1 (Feb. 2000), pp. 196–201. DOI: 10.1109/59.852121.

[KB00] B. Kim and R. Baldick. “A Comparison of Distributed Optimal Power Flow Algorithms”. In: *IEEE Transactions on Power Systems* 15.2 (May 2000), pp. 599–604. DOI: 10.1109/59.867147.

[WB00] A. Wächter and L. T. Biegler. “Failure of Global Convergence for a Class of Interior Point Methods for Nonlinear Programming”. In: *Mathematical Programming* 88.3 (Sept. 2000), pp. 565–574. DOI: 10.1007/PL00011386.

[AQC99] J. Aguado, V. Quintana, and A. Conejo. “Optimal Power Flows of Interconnected Power Systems”. In: *1999 IEEE Power Engineering Society Summer Meeting. Conference Proceedings*. Vol. 2. July 1999, 814–819 vol.2. DOI: 10.1109/PESS.1999.787421.

[Bal+99] R. Baldick, B. Kim, C. Chase, and Y. Luo. “A Fast Distributed Implementation of Optimal Power Flow”. In: *IEEE Transactions on Power Systems* 14.3 (Aug. 1999), pp. 858–864. DOI: 10.1109/59.780896.

[Ber99] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, 1999.

[LL99] S. Low and D. Lapsley. "Optimization Flow Control. I. Basic Algorithm and Convergence". In: *IEEE/ACM Transactions on Networking* 7.6 (Dec. 1999), pp. 861–874. DOI: 10.1109/90.811451.

[Ber98] D. P. Bertsekas. *Network Optimization: Continuous and Discrete Methods*. Athena Scientific, 1998.

[CA98] A. Conejo and J. Aguado. "Multi-Area Coordinated Decentralized DC Optimal Power Flow". In: *IEEE Transactions on Power Systems* 13.4 (Nov. 1998), pp. 1272–1278. DOI: 10.1109/59.736264.

[KMT98] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan. "Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability". In: *Journal of the Operational Research Society* 49.3 (Mar. 1998), pp. 237–252. DOI: 10.1057/palgrave.jors.2600523.

[Roc98] R. T. Rockafellar. *Network Flows and Monotropic Optimization*. Athena Scientific, 1998.

[CZ97] Y. Censor and S. A. Zenios. *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press, 1997.

[Mom+97] J. Momoh, R. Koessler, M. Bond, B. Stott, D. Sun, A. Papalexopoulos, and P. Ristanovic. "Challenges to Optimal Power Flow". In: *IEEE Transactions on Power Systems* 12.1 (Feb. 1997), pp. 444–455. DOI: 10.1109/59.575768.

[Wri97] S. J. Wright. *Primal-Dual Interior-Point Methods*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, Jan. 1997. DOI: 10.1137/1.9781611971453.

[BT95] P. T. Boggs and J. W. Tolle. "Sequential Quadratic Programming". In: *Acta numerica* 4 (1995), pp. 1–51.

[GS94] J. J. Grainger and W. D. Stevenson. *Power System Analysis*. McGraw-Hill, 1994.

[LHA94] F. Lee, J. Huang, and R. Adapa. "Multi-Area Unit Commitment via Sequential Method and a DC Power Flow Network Model". In: *IEEE Transactions on Power Systems* 9.1 (Feb. 1994), pp. 279–287. DOI: 10.1109/59.317600.

[Tsi94] J. N. Tsitsiklis. "Asynchronous Stochastic Approximation and Q-Learning". In: *Machine Learning* 16.3 (Sept. 1994), pp. 185–202. DOI: 10.1007/BF00993306.

[HO93] G. Huang and W. Ongsakul. “Managing the Bottlenecks in Parallel Gauss-Seidel Type Algorithms for Power Flow Analysis”. In: *Conference Proceedings Power Industry Computer Application Conference*. May 1993, pp. 74–81. DOI: 10.1109/PICA.1993.291033.

[SVN93] Shyan-Lung Lin and J. Van Ness. “Parallel Solution of Sparse Algebraic Equations”. In: *Conference Proceedings Power Industry Computer Application Conference*. May 1993, pp. 380–386. DOI: 10.1109/PICA.1993.290992.

[Com92] P. S. E. Committee. “Parallel Processing in Power Systems Computation”. In: *IEEE Transactions on Power Systems* 7.2 (May 1992), pp. 629–638. DOI: 10.1109/59.141768.

[EB92] J. Eckstein and D. P. Bertsekas. “On the Douglas—Rachford Splitting Method and the Proximal Point Algorithm for Maximal Monotone Operators”. In: *Mathematical Programming* 55.1 (Apr. 1992), pp. 293–318.

[ETA90] M. Enns, W. Tinney, and F. Alvarado. “Sparse Matrix Inverse Factors (Power Systems)”. In: *IEEE Transactions on Power Systems* 5.2 (May 1990), pp. 466–473. DOI: 10.1109/59.54554.

[FI90] A. V. Fiacco and Y. Ishizuka. “Sensitivity and Stability Analysis for Nonlinear Programming”. In: *Annals of Operations Research* 27.1 (Dec. 1990), pp. 215–235. DOI: 10.1007/BF02055196.

[OKS90] T. Oyama, T. Kitahara, and Y. Serizawa. “Parallel Processing for Power System Analysis Using Band Matrix”. In: *IEEE Transactions on Power Systems* 5.3 (Aug. 1990), pp. 1010–1016. DOI: 10.1109/59.65932.

[BW89] M. Baran and F. Wu. “Optimal Sizing of Capacitors Placed on a Radial Distribution System”. In: *IEEE Transactions on Power Delivery* 4.1 (Jan. 1989), pp. 735–743. DOI: 10.1109/61.19266.

[BT89] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Vol. 23. Prentice Hall Englewood Cliffs, NJ, 1989.

[MPG87] A. Monticelli, M. V. F. Pereira, and S. Granville. “Security-Constrained Optimal Power Flow with Post-Contingency Corrective Rescheduling”. In: *IEEE Transactions on Power Systems* 2.1 (Feb. 1987), pp. 175–180. DOI: 10.1109/TPWRS.1987.4335095.

[SAY87] H. Sasaki, K. Aoki, and R. Yokoyama. “A Parallel Computation Algorithm for Static State Estimation by Means of Matrix Inversion Lemma”. In: *IEEE Transactions on Power Systems* 2.3 (Aug. 1987), pp. 624–631. DOI: 10.1109/TPWRS.1987.4335182.

[BA86] R. Betancourt and F. L. Alvarado. “Parallel Inversion of Sparse Matrices.” In: *IEEE Transactions on Power Systems* 1.1 (Feb. 1986), pp. 74–81. DOI: 10.1109/TPWRS.1986.4334846.

[TBA86] J. Tsitsiklis, D. Bertsekas, and M. Athans. “Distributed Asynchronous Deterministic and Stochastic Gradient Optimization Algorithms”. In: *IEEE Transactions on Automatic Control* 31.9 (Sept. 1986), pp. 803–812. DOI: 10.1109/TAC.1986.1104412.

[Ber83] D. P. Bertsekas. “Distributed Asynchronous Computation of Fixed Points”. In: *Mathematical Programming* 27.1 (Sept. 1983), pp. 107–120. DOI: 10.1007/BF02591967.

[Gab83] D. Gabay. “Applications of the Method of Multipliers to Variational Inequalities”. In: *Studies in Mathematics and Its Applications*. Vol. 15. Elsevier, 1983, pp. 299–331.

[Nes83] Y. Nesterov. “A Method for Solving the Convex Programming Problem with Convergence Rate O($1/K^2$)”. In: *Dokl. Akad. Nauk Sssr*. Vol. 269. 1983, pp. 543–547.

[TPG83] S. N. Talukdar, S. S. Pyo, and T. C. Giras. “Asynchronous Procedures for Parallel Processing”. In: *IEEE Transactions on Power Apparatus and Systems* PAS-102.11 (Nov. 1983), pp. 3652–3659. DOI: 10.1109/TPAS.1983.317728.

[VNM83] J. E. Van Ness and G. Molina. “The Use of Multiple Factoring in the Parallel Solution of Algebraic Equations”. In: *IEEE Transactions on Power Apparatus and Systems* PAS-102.10 (Oct. 1983), pp. 3433–3438. DOI: 10.1109/TPAS.1983.317840.

[Ber82] D. P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, 1982.

[DES82] R. S. Dembo, S. C. Eisenstat, and T. Steihaug. “Inexact Newton Methods”. In: *SIAM Journal on Numerical analysis* 19.2 (1982), pp. 400–408.

[WHB81] Y. Wallach, E. Handschin, and C. Bongers. “An Efficient Parallel Processing Method for Power System State Estimation”. In: *IEEE Transactions on Power Apparatus and Systems* PAS-100.11 (Nov. 1981), pp. 4402–4406. DOI: 10.1109/TPAS.1981.316852.

[HS80] W. Hock and K. Schittkowski. “Test Examples for Nonlinear Programming Codes”. In: *Journal of optimization theory and applications* 30.1 (1980), pp. 127–129.

[Rob80] S. M. Robinson. “Strongly Regular Generalized Equations”. In: *Mathematics of Operations Research* 5.1 (1980), pp. 43–62.

[FP78] J. Fong and C. Pottle. “Parallel Processing of Power System Analysis Problems Via Simple Parallel Microcomputer Structures”. In: *IEEE Transactions on Power Apparatus and Systems* PAS-97.5 (Sept. 1978), pp. 1834–1841. DOI: 10.1109/TPAS.1978.354677.

[GM76] D. Gabay and B. Mercier. “A Dual Algorithm for the Solution of Nonlinear Variational Problems via Finite Element Approximation”. In: *Computers & Mathematics with Applications* 2.1 (1976), pp. 17–40. DOI: 10.1016/0898-1221(76)90003-1.

[GM75] R. Glowinski and A. Marroco. “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 (1975), pp. 41–76.

[Mie+72] A. Miele, P. E. Moseley, A. V. Levy, and G. M. Coggins. “On the Method of Multipliers for Mathematical Programming Problems”. In: *Journal of Optimization Theory and Applications* 10.1 (July 1972), pp. 1–33.

[Mie+71] A. Miele, E. E. Cragg, R. R. Iyer, and A. V. Levy. “Use of the Augmented Penalty Function in Mathematical Programming Problems, Part 1”. In: *Journal of Optimization Theory and Applications* 8.2 (Aug. 1971), pp. 115–130.

[Hes69] M. R. Hestenes. “Multiplier and Gradient Methods”. In: *Journal of Optimization Theory and Applications* 4.5 (Nov. 1969), pp. 303–320.

[Pow69] M. Powell. *A Method for Nonlinear Constraints in Minimization Problems*. Optimization. Academic Press, 1969.

[Eve63] H. Everett. “Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources”. In: *Operations research* 11.3 (1963), pp. 399–417.

[Ben62] J. F. Benders. “Partitioning Procedures for Solving Mixed-Variables Programming Problems”. In: *Numerische Mathematik* 4.1 (Dec. 1962), pp. 238–252.

[DW60] G. B. Dantzig and P. Wolfe. “Decomposition Principle for Linear Programs”. In: *Operations research* 8.1 (1960), pp. 101–111.

[Bel58] R. Bellman. “On a Routing Problem”. In: *Quarterly of Applied Mathematics* 16.1 (1958), pp. 87–90. DOI: 10.1090/qam/102435.

[Neu49] J. V. Neumann. “On Rings of Operators. Reduction Theory”. In: *Annals of Mathematics* (1949), pp. 401–485.

Publications

Journal Articles

[Bot+20] M. Botkin-Levy, A. Engelmann, T. Mühlpfordt, T. Faulwasser, and M. Almassalkhi. “Distributed Control of Charging for Electric Vehicle Fleets under Dynamic Transformer Ratings”. In: *submitted to IEEE Transactions on Control Systems Technology* (July 2020). arXiv: 2007.10304.

[Eng+20] A. Engelmann, H. Benner, R. Ou, Y. Jiang, B. Houska, and T. Faulwasser. “ALADIN-Alpha—An Open-Source MATLAB Toolbox for Distributed Non-Convex Optimization”. In: *submitted to Optimal Control Applications and Methods* (2020). arXiv: 2006.01866.

[Eng+20] A. Engelmann, Y. Jiang, B. Houska, and T. Faulwasser. “Decomposition of Non-Convex Optimization via Bi-Level Distributed ALADIN”. In: *IEEE Transactions on Control of Network Systems* (2020). DOI: 10.1109/TCNS.2020.3005079.

[Eng+19b] A. Engelmann, Y. Jiang, T. Mühlpfordt, B. Houska, and T. Faulwasser. “Toward Distributed OPF Using ALADIN”. In: *IEEE Transactions on Power Systems* 34.1 (Jan. 2019), pp. 584–594. DOI: 10.1109/TPWRS.2018.2867682.

[FE19] T. Faulwasser and A. Engelmann. “Toward Economic NMPC for Multistage AC Optimal Power Flow”. In: *Optimal Control Applications and Methods* 41.1 (2019), pp. 107–127. DOI: 10.1002/oca.2487.

[Fau+18] T. Faulwasser, A. Engelmann, T. Mühlpfordt, and V. Hagenmeyer. “Optimal power flow: an introduction to predictive, distributed and stochastic control challenges”. In: *at - Automatisierungstechnik* 66.7 (July 2018), pp. 573–589. DOI: 10.1515/auto-2018-0040.

Conference Papers

[Du+20] X. Du, A. Engelmann, Y. Jiang, T. Faulwasser, and B. Houska. “Optimal Experiment Design for AC Power Systems Admittance Estimation”. In: *Proceedings of the 21th IFAC World Congress, Berlin*. 2020.

[Du+19] X. Du, A. Engelmann, Y. Jiang, T. Faulwasser, and B. Houska. “Distributed State Estimation for AC Power Systems Using Gauss-Newton ALADIN”. In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. Dec. 2019, pp. 1919–1924. DOI: 10.1109/CDC40024.2019.9028966.

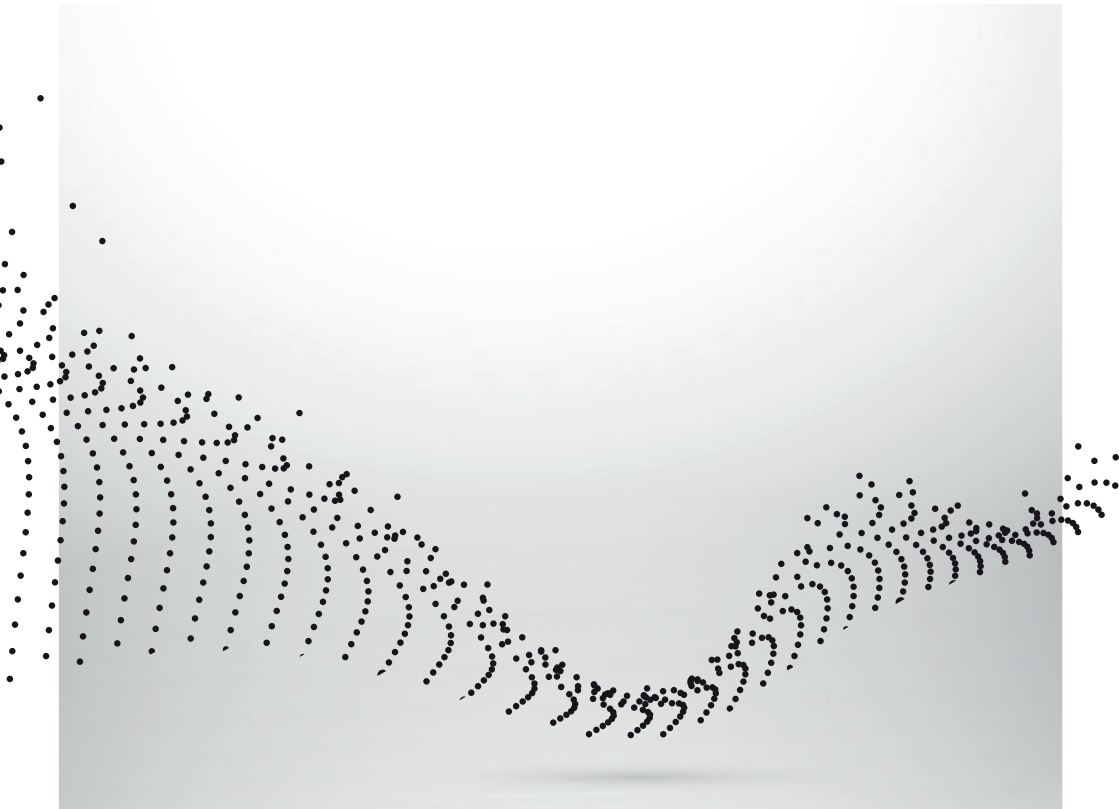
[Zwi+19] P. Zwickel, A. Engelmann, L. Gröll, V. Hagenmeyer, D. Sauer, and T. Faulwasser. “A Comparison of Economic MPC Formulations for Thermal Building Control”. In: *2019 IEEE PES Innovative Smart Grid Technologies Europe*. Sept. 2019, pp. 1–5. DOI: 10.1109/ISGTEurope.2019.8905593.

[EF18] A. Engelmann and T. Faulwasser. “Feasibility vs. Optimality in Distributed AC OPF - A Case Study Considering ADMM and ALADIN”. In: *Proceedings of the Second International Symposium on Energy System Optimization*. 2018. DOI: 10.1007/978-3-030-32157-4_1.

[Eng+18a] A. Engelmann, T. Mühlpfordt, Y. Jiang, B. Houska, and T. Faulwasser. “Distributed Stochastic AC Optimal Power Flow Based on Polynomial Chaos Expansion”. In: *2018 Annual American Control Conference (ACC)*. June 2018, pp. 6188–6193. DOI: 10.23919/ACC.2018.8431090.

[Mur+18] A. Murray, A. Engelmann, V. Hagenmeyer, and T. Faulwasser. “Hierarchical Distributed Mixed-Integer Optimization for Reactive Power Dispatch”. In: *IFAC-PapersOnLine* 51.28 (2018), pp. 368–373. DOI: 10.1016/j.ifacol.2018.11.730.

[Eng+17] A. Engelmann, T. Mühlpfordt, Y. Jiang, B. Houska, and T. Faulwasser. “Distributed AC Optimal Power Flow Using ALADIN”. In: *IFAC-PapersOnLine*. Vol. 50. 20th IFAC World Congress. July 2017, pp. 5536–5541. DOI: 10.1016/j.ifacol.2017.08.1095.



In many engineering domains, systems are composed of partially independent subsystems – power systems are composed of distribution and transmission systems, teams of robots are composed of individual robots, and chemical process systems are composed of vessels, heat exchangers and reactors. Often, these subsystems should reach a common goal such as satisfying a power demand with minimum cost, flying in a formation, or reaching an optimal set-point.

Mathematical optimization techniques are among the most successful tools for controlling such systems optimally with feasibility guarantees. Yet, they are often centralized – all data has to be collected in one central and computationally powerful entity. Methods from distributed optimization overcome this limitation. Classical approaches, however, are often not applicable due to non-convexities. The present work develops one of the first frameworks for distributed optimization with non-convex constraints.

ISBN 978-3-7315-1180-9

Gedruckt auf FSC-zertifiziertem Papier