Distributed Convex Optimization
Based on Monotone Operator Theory

Sherson, Thomas

DOI
10.4233/uuid:fb60dba0-e5f9-451e-b664-e3ca0d45b36b

Publication date
2019

Document Version
Final published version

Citation (APA)
Sherson, T. (2019). *Distributed Convex Optimization: Based on Monotone Operator Theory*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:fb60dba0-e5f9-451e-b664-e3ca0d45b36b

DISTRIBUTED CONVEX OPTIMIZATION

BASED ON MONOTONE OPERATOR THEORY

DISTRIBUTED CONVEX OPTIMIZATION

BASED ON MONOTONE OPERATOR THEORY

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof.dr.ir. T.H.J.J. van der Hagen,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op maandag 3 juni 2019 om 12:30 uur

door

Thomas William SHERSON

Bachelor of Engineering with Honors, Victoria University of Wellington, New Zealand
geboren te Petersfield, Verenigd Koninkrijk.

Dit proefschrift is goedgekeurd door de

Promotor: Prof.dr.ir. W.B. Kleijn
Promotor: Dr. ir. R. Heusdens

Samenstelling promotiecommissie:

Rector Magnificus,	voorzitter
Prof. dr. ir. W.B. Kleijn,	Technische Universiteit Delft, Netherlands Victoria University of Wellington, New Zealand,
Dr. ir. R. Heusdens,	Technische Universiteit Delft, Netherlands
Onafhankelijke leden:	
Dr. F.M. de Oliveria Filho	Technische Universiteit Delft, Netherlands
Prof. dr. ir. M. Moonen	Katholieke Universiteit Leuven, Belgium
Prof. dr. C. Richard	Université de Nice Sophia-Antipolis, France
Prof. dr. ir. M. Verhaegen	Technische Universiteit Delft, Netherlands
Prof. dr. ir. A.J. van der Veen	Technische Universiteit Delft, Netherlands, reservelid

This research was funded as part of the "Distributed Processing of Audio Signals" project sponsored by Huawei.

Keywords: Distributed Signal Processing, Convex Optimization, Monotone Operator Theory, Wireless Sensor Networks

Printed by: Ipskamp Printing

Front & Back: Ruby Urquhart

ISBN 978-94-6384-041-5

An electronic version of this dissertation is available at
http://repository.tudelft.nl/.

CONTENTS

I

PROLOGUE

1

INTRODUCTION

"You don't have to be a fantastic hero to do certain things, to compete. You can be just an ordinary chap, sufficiently motivated to reach challenging goals. The intense effort, the giving of everything you've got, is a very pleasant bonus"

Edmund Hillary

1

1.1. OVERVIEW

The focus of this thesis is the analysis and design of various solvers for use in distributed convex optimization. Motivated by the inherent link between signal processing and convex problems, the design of such solvers aims to facilitate the implementation of distributed signal processing algorithms in adhoc and large scale networks without the need for packet passing or data aggregation. In particular, we approach the task of this design process from the perspective of monotone operator theory which provides a unifying perspective of many different first order convex solvers. In this initial chapter we provide the contextual basis for this work by reflecting on the role of networking within current society. We also provide an outline of the remainder of this thesis and our contributions to the field of distributed optimization.

1.2. MOTIVATION: COMPUTING IN A NETWORKED WORLD

One of the hallmarks of the living world is the ability of members of a species to collaboratively work together to achieve a common goal. Be it a pack of lions hunting gazelle, a flock of geese flying in a V-formation to reduce air resistance, or a shoal of fish swimming together to reduce their possibility of being eaten, collaboration is an essential component to survival. The rise of humanity has also been inherently dependent on our ability to work together. From the way we adopt complementary rolls in a society, through to sharing ideas with one another, our ability to communicate and collaborate has driven our success. A similar story is reflected in the world of computing. Since their conception at the end of the first half of the twentieth century, the paradigm of computing has transformed from a landscape of isolated and disconnected entities through to a sprawling global web of interconnected devices.

Fast forward fifty years and in response to our ability to coordinate computers over distances both short and long we have seen rapid advances in the utilities and services that underpin out modern world. From the way that we share information via the internet, to our interactions via social media [1], through to the way we store and process data via cloud based services [2], and to more fundamental tasks such as power distribution (e.g. smart-grid power networks, demand side management [3]) and transportation (e.g. autonomous fleet navigation [4]), networking is playing an increasingly central role in many facets of our lives.

Figure 1.1: Agent collaboration in nature and society. On the left a group of geese are flying in formation to reduce are drag. On the right a swarm of drones are flying in formation as part of light show.

In parallel to the networking of more traditional computers, the emergence of the "Internet of Things" (IoT) within the last few years has been driving the ubiquity of low cost interconnected devices to new heights [5]. By the year 2025 for instance, it is predicted that more than 75 billion wireless equipped devices will be in active deployment, a more than six fold increase since the estimated number of connected devices in 2012 [6]. Combined with the ever growing coverage of wireless communication platforms across the world and the increasing computational capabilities of such devices, riding on the crest of Moore's law, everything from desktop computers, to cellphones, to home appliances and even disposable low cost sensors can form part of a growing sea of networked computers.

The fore-coming of a massively networked world and the plethora of information it could capture offers interesting opportunities for us as signal processing engineers to take advantage of. Be it traffic congestion tracking [7], intra-city weather detection[8], air pollution monitoring [9] and more, a highly interconnected society provides an ideal platform for new and innovative solutions for the modern world. Such tasks are often labeled as "Big Data" problems, a name adopted due to the sheer scale of data which is often available for processing, but this blanket term neglects an important feature of such data sets; this information stems from a network of computational units. Therefore, while we could process such data using a single super computer, a more interesting question is whether a network can be made to work together to achieve the same feat. Thus, just as a species must learn effective strategies to achieve a common goal, the omnipotence of networked systems in our lives necessitates the design of special algorithms to take full advantage of their capabilities. As computers lack the cognition to design such methods for themselves (at least given the current state of play) it is our role as engineers to address this task. We must be the ones to devise effective strategies to the task of signal processing in networks and we must do so whilst simultaneously making the most of what these systems have to offer as well as ensuring that we respect their limitations.

At its heart, this thesis, explores the following problem:

Question 1. *How can we design methods to allow computers to work together to achieve a common goal in a landscape of networked devices?*

In particular, we explore this question from the perspective of *distributed signal processing* and its relationship with *distributed optimization*. The importance of this link and in turn the specific focus of this thesis is introduced in the following sections.

1.3. WHY DISTRIBUTED SIGNAL PROCESSING?

Historically, signal processing is a task which is performed by a single machine. Crudely speaking, data is collected, be it by a physical sensor or generated artificially, before being transmitted to a central location at which point we can apply our favorite tools be it filtering, data transformation, clustering or more. Such systems are attractive as they have a simple architecture and hierarchy (there is one master compute node while all other sensor nodes act as slaves), all of the information exists at a single location, and ultimately they are familiar to work with. However, in the world of networked systems and Big Data, these classical topologies are not without their faults. Perhaps the simplest

drawback of such systems is scalability. As network size increases, the amount of data generated by all of the sensors can increase dramatically necessitating the storage and processing capabilities of the central node to increase in turn. For real world systems with rapidly increasing number of nodes, this type of complexity scaling is unsustainable. Similarly, these networks offer little robustness to system failures as they exhibit a single point of failure. Should the central processing point fail for any reason, the entire process is compromised. To circumvent these limitations we must turn our attention to other approaches.

Distributed signal processing aims to address the limitations of classical centralized systems by directly exploiting the localized nature of generated data, i.e., that each node in the network is associated with a subset of the overall data. If we can allow each node to store this information, rather than aggregating it to a central point, the memory capabilities of the network would scale with the number of nodes. Similarly, as each node has some form of computational capabilities, if we can also partition any computation over the set of nodes as well, the compute power of our network will scale with the number of nodes. In such a paradigm we have also removed the hierarchy that perviously existed in a centralized system by removing any dependence on a master computer. An example, contrasting the network topologies of these centralized and distributed networks is demonstrated in Figure 1.2.

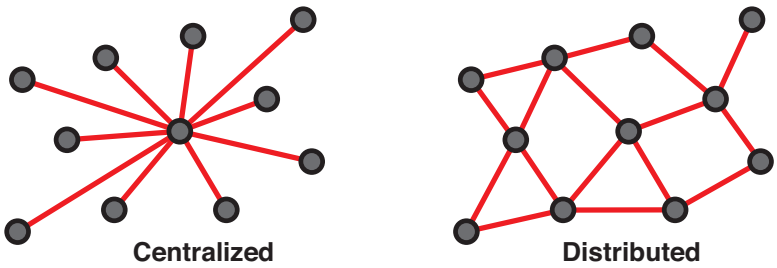


Figure 1.2: A comparison of the network structure of a centralized and distributed network. The gray circles denote nodes in the network while the red lines denote communication channels between nodes.

In a distributed network, as every node takes part in both data storage and data processing, the resulting system is also fundamentally more robust in the face of system failures due to the absence of any single points of failure. However, the biggest challenge for such systems becomes the actual implementation of the desired signal processing operations in such a context. As no one node has access to all the information in the network, even simple operations such as computing inner products become infeasible without the use of data aggregation, the introduction of which essentially reduces the distributed nature of the network back to that a centralized form.

To overcome address the limitation of restricted data locality, a name which reflects the naturally localized nature of data within the network, we must let the nodes exchange information. However, rather than letting nodes aggregate data across the entire network, in distributed signal processing, we impose that nodes must only share information with each other only if they can directly communicate. For a given node, those other nodes with which it can communicate are referred to as its neighbors. An example

of such a neighborhood is given below in Figure 1.3.

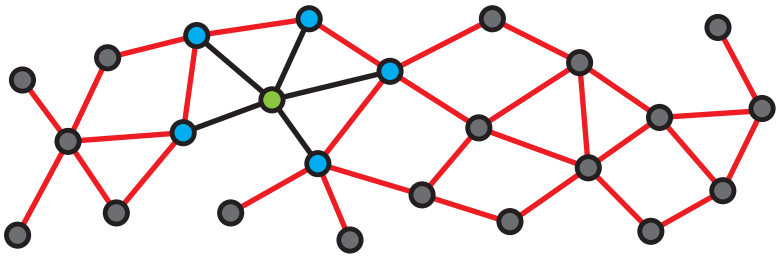


Figure 1.3: An example of the set of neighbors of a node in a distributed network. The set of blue nodes are the neighbors of the green node in this instance with the associated communication channels colored in black.

In practical contexts, the aforementioned restriction has two benefits. Firstly, if the topology of a network is geographically generated, neighboring nodes will be closer together and thus exchanging data with each other will require less transmission power than sharing information with other distant nodes. Secondly, by naturally restricting the number of other nodes with which a given node can communicate, the amount of data any one node may need to store need not increase with size network, preserving the appealing scalability property introduced above. While in some contexts, allowing this limited form of communication may be sufficient to perform some signal processing operations, in general we require additional methods to develop a truly general distributed signal processing platform.

Within the literature a myriad of existing approaches have been proposed to perform distributed computation including the likes of distributed consensus methods (distributed subgradient descent [10], subgradient-push optimization [11], randomized gossip [12]), message passing or belief propagation (max-sum method [13], sum-product [14], loopy belief propagation [15]), graph filtering (distributed FIR filtering [16], distributed ARMA filtering [17]) and more. Of additional interest to this work is the field of distributed convex optimization which includes the likes of the alternating direction method of multiplier (ADMM) [18], ADMM+ [19], AFBA [20] and more as special cases. In the following we motivate why this approach represents an attractive option for distributed signal processing.

1.4. DISTRIBUTED CONVEX OPTIMIZATION

In parallel to the emergence of hugely networked systems, the field of signal processing has seen a rapid uptake of optimization methods in recent years. Following the development of powerful solvers, such as the interior point method using self concordant barrier functions as proposed by Nesterov and Nemirovski [21], a plethora of traditional signal processing problem have been rephrased as equivalent convex optimization problems. Notably, the generality and flexibility of optimization has seen its emergence as a defacto approach for a wide range of applications in telecommunications [22], acoustic signal processing [23], control theory [24], image processing [25] and more. In the context of this thesis, the synergy between convex optimization and signal processing offers an attractive stepping stone towards the goal of deriving distributed signal processing

methods. Specifically, to circumvent the need for designing dedicated signal processing solutions for a given application, we can instead develop distributed solvers for convex optimization problems therefore reducing algorithm design to the more familiar area of problem transformation.

The notion of using convex optimization in the context of distributed signal processing is not itself new with a lineage dating back to the late 1970's and early 1980's [26, 27, 28, 29, 30, 31, 32, 33]. In recent years however, driven by the explosion in the use of networked systems, such applications have received wide spread attention in the literature with a wide range of approaches being proposed as a result. While these approaches take on a variety of forms, they all share the common goal introduced above, that for a network to effectively work the elements or nodes of network must combine their own local computational capabilities with their ability to communicate with each other to solve a given task. In other words, a given optimization problem should be solved through a combination of local operations at each node in the network and an exchange of information between connected devices. The hope is then that by repeatedly alternating between these operations, the network can jointly solve a given task.

To complement the plethora of existing research within the literature and to ultimately address Question 1, in this thesis we explore three main branches of research; improving the understanding and analysis of existing distributed solvers, proposing new solvers to broaden the class of problems which can be solved in a distributed manner and finally demonstrating the use of such methods in a practical distributed signal processing context. These three areas are discussed in more detail in the following subsections.

1.4.1. ANALYSIS OF EXISTING DISTRIBUTED SOLVERS

The first portion of this thesis, focuses on understanding the performance of existing algorithms for use in distributed optimization, answering questions such as how quickly can an algorithm find an optimal solution, for which families of problems can an algorithm converge and more. Unfortunately, the literature contains a broad spectrum of convex solvers all with seemingly disparate derivations. This makes the analysis and understanding of different algorithms a challenging task as each necessitates specifically tailored tools to verify its performance. Thankfully there exists a general mathematical framework, named monotone operator theory [34], through which many first order convex solvers can be derived. An overview of this framework along with the relevant properties for this thesis is provided in Chapter 2. Such a framework, which is highly mature and well understood, represents a powerful tool for understanding the convergence characteristics of existing distributed solvers from within the literature.

One such target for this analysis is the primal dual method of multipliers (PDMM), a recently proposed distributed solver whose existing derivation was disjoint from other approaches in the literature. Specifically while PDMM offers appealing performance in empirical testing, its theoretical guarantees were limited at best, prior to our research, due its atypical formulation. The first research question of this thesis was therefore as follows:

Question 2. *How does PDMM relate to other distributed solvers within the literature and for what types of problems can guarantee convergence?*

Namely, we wanted to demonstrate that PDMM could be unified with other approaches within the literature by re-deriving it from the perspective of monotone operator theory. Furthermore, if such a connection were to be made, we could then utilize the extensive results from monotone operator theory to strengthen the convergence results for the method and to offer insight into its theoretical performance guarantees.

Monotone operator theory can also be applied to the complementary task of understanding how the topology of a given network can influence the convergence characteristics of a given solver. Specifically, in the case of PDMM we were curious if a connection could be drawn between algorithmic convergence rate and the given connectivity between nodes. This led to our second research question for this thesis:

Question 3. *How does network topology affect the convergence of PDMM and can this impact be quantified?*

Answering this question would allow designers to make informative decisions between different network topologies as well as to better understand how parameter selection may influence the performance of a given algorithm.

1.4.2. DESIGNING DISTRIBUTED SOLVERS

The second area of focus for this thesis was that of solver design. The design of optimization solvers is a classic task within the field of computer science with the main objective being to identify how to exploit the specific structure of a class of problems to develop computationally efficient algorithms. Throughout the 20th century, the development of methods such as Dantzig's simplex approach to solving linear programs [35], Khachiyan's interior point method for linear programming [36] (the first ever polynomial time for this problem class) or the aforementioned log barrier interior point methods for semi-definite programming revolutionized the field of applied mathematics. However, driven by the need for optimization tools in Big Data applications, the last decade and a half has a seen push towards computationally efficient methods for performing optimization on a scale essentially unreachable by more general tools. As such problems are typically solved in a centralized fashion, these algorithms often exploit stochastic update procedures, subgradient methods and even problem approximation, all in an effort to reduce the computational complexity of any one iteration. Similarly, solvers for distributed optimization problems impose their own restrictions, most notably that their implementations naturally lead to the parallelization of operations between the nodes. As previously, mentioned, the last decade has seem a dramatic increase in the variety of solvers available [18, 37, 38, 19, 20] including everything from traditional approaches such as ADMM through to novel state of the art methods.

Given the range of existing distributed solvers within the literature, it begs the question why focus on developing new solvers? From a distributed signal processing perspective, the choice between different solvers relates to the basic prototype problem they can be used to address. Specifically, when reducing distributed signal processing to that of problem transformation, recasting a desired operation as an convex optimization problem, the generality of a given prototype problem is of upmost importance. Notable, the more general the prototype problem, the easier it may be to convert a desired signal processing approach to a distributed form. For this reason, the focus of this branch of our

research was to broaden the types of problems we can solve in a distributed manner. In particular, we were curious if we could define sufficient characteristics of problems for which we could find a distributed solver ultimately leading to the following research question:

Question 4. *Is problem separability a sufficient condition for distributed optimization and if so how can it be exploited?*

A related area of distributed solver design focuses on where we allow for more natural characteristics of the networks with which we want to work with. For instance, if we build a network out of the cellphones of people walking within a city, the topology of our network will vary with time. While there are methods within the literature for performing distributed optimization in such networks [39, 40, 11], these often restrict the types of problems for which convergence can be guaranteed. We were therefore interested in the following research question:

Question 5. *Does a time varying network topology still facilitate distributed optimization of general convex problems and can solvers be derived for such a context?*

Specifically, we wanted to utilize the same monotone operator framework used in the analysis of existing algorithms in this formulation process. Such an approach would again allow us to take advantage of the wealth of results offered by this framework to develop an understanding of the theoretical performance of any proposed methods.

1.4.3. DISTRIBUTED SIGNAL PROCESSING IN PRACTICE

The final research direction considered in this thesis is that of applying the theoretical methods developed herein in solving practical signal processing problems. The importance of this work is that it demonstrates both an exploration of the practical use of such methods but also the considerations that must be made during the problem reformulation stage of a distributed implementation. In particular, we considered an application from audio signal processing. Given the ubiquity of cellphones in modern society which, with their multiple microphones and wireless capabilities, could be made to form a wireless acoustic sensor network (WASN), we were interested in seeing if the proposed distributed optimization methods could be used for realtime acoustic signal processing. A natural task for distributed acoustic signal processing is that of multichannel channel noise reduction where we essentially want to combine the observations of a target signal from multiple microphones (cellphones in this context) to improve the quality of a target source. This led to the fifth question for this thesis:

Question 6. *How can we perform beamforming within a distributed network of wireless acoustic sensor nodes?*

Specifically, we explored not only the theoretical requirements for implementing such a method via distributed optimization, but also the practical considerations unique to an audio signal processing context and how these can be accommodated for in a distributed environment.

1.5. CONTRIBUTIONS AND THESIS OUTLINE

Based on the three branches of research introduced above, the main contribution of this thesis can be summarized as an exploration of the different ways in which monotone operator theory can be used to expand the scope of distributed convex optimization. To support in this pursuit, Chapter 2 provides a general overview of monotone operator theory and its relation to convex optimization. In particular, we demonstrate how monotone operator theory provides a unifying means of accessing the performance of different first order solvers. Furthermore we motivate the key objectives of distributed optimization and provide a simple example problem solved by the well known alternating direction method of multipliers (ADMM).

The subsequent chapters are separated based on the three research branches and explore how this theory can be applied to these different aspects of distributed signal processing. Chapter 3, for instance, demonstrates how the primal dual method of multipliers (PDMM), which was recently proposed as a new method for distributed optimization, can be derived from the perspective of monotone operator theory. In contrast to efforts within the literature, this provides a concrete link between PDMM and other existing approaches such as ADMM and allows us to demonstrate stronger convergence results such as sufficient conditions for convergence and, under stronger functional assumptions, geometric convergence as well.

Chapter 4 acts as a supplement to Chapter 3 and demonstrates a modified PDMM algorithm which, unlike its unmodified form, guarantees convergence for all closed, convex and proper functions. In particular, by incorporating an additional primal regularization step, which is shown to be derivable from the perspective of monotone operator theory, the resulting method attains the same node-based structure of PDMM while improving the convergence guarantees of the algorithm.

In Chapter 5 we change directions slightly and focus on analyzing the effect of network topology on the convergence rate of PDMM. Notably, for such problems that PDMM converges at a geometric rate, we demonstrate how this rate is parameterized by the random walk rate of the underlying graph. This result not only provides us with an analytic worst case convergence rate bound but also demonstrates an inherent link between a distributed convex optimization algorithm and spectral graph theory. Using this result we are able to then derive convergence characteristics for a number of deterministic graphs and specifically to demonstrate a problem for which PDMM converges in a finite number of iterations.

Chapter 6 then focuses on the task of broadening the class of problems which can be solved in a distributed manner. By constructing a particular dual lifted problem, the proposed distributed method of multipliers (DMM) can be used to solve general separable optimization problems, those with separable objectives and constraints) and can do so in an entirely distributed fashion. We show how this approach can be derived from the perspective of monotone operator theory and demonstrate its use in a number of distributed signal processing applications.

Chapter 7 also focuses on a task relating to network topology by demonstrating a novel algorithm for use in time varying networks. Specifically, the proposed method, which again stems from monotone operator theory, exploits a time varying choice of metric in conjunction with a clever reformulation of the update equations to construct

an algorithm whose convergence does not depend on network topology. In particular, we provide guaranteed convergence for a range of functions and also highlight more general functional classes for which the method still works effectively.

Chapter 8 provides an example application of applying PDMM to a real signal processing problem, that of acoustic beamforming. In this work, we present a novel distributed beamformer for use in wireless acoustic sensor networks, and demonstrate how the proposed signal processing problem can be implemented using PDMM. The proposed method offers improved robustness to steering vector mismatch whilst being entirely distributable. Furthermore, we demonstrate how a warm start procedure can be used to reduce the number of iterations required by the system while maintaining a high level of performance.

Finally, in Chapter 9 we provide our concluding remarks of this thesis and highlight potential avenues for future work in the field of distributed optimization based on monotone operator theory.

The general flow of the chapters is given as follows:

1. Chapter 2 introduces appropriate background information on monotone operator theory and its application to convex optimization.

2. Chapter 3 demonstrate how PDMM can be derived from the perspective of monotone operator theory and uses this link to demonstrate new convergence results for the method.

3. Chapter 4 demonstrates a simple modification for PDMM, again based on monotone operator theory, which can be used to guarantee algorithmic convergence for a broader class of objective functions.

4. Chapter 5 analyzes the effect of network topology on the convergence rate of PDMM and links this rate with results from spectral graph theory

5. Chapter 6 demonstrates a novel algorithm for distributed optimization, allowing for general separable problems to be solved in a fully distributed manner.

6. Chapter 7 demonstrates a novel algorithm for distributed consensus in time varying networks, again based on monotone operator theory.

7. Chapter 8 highlights the use of PDMM in an acoustic signal processing application through the development of a novel beamforming algorithm for use in wireless acoustic sensor networks.

8. Chapter 9 provides our concluding remarks and comments for future extensions of this work.

1.6. LIST OF PUBLICATIONS AND OTHER CONTRIBUTIONS

LIST OF JOURNALS

1. **Thomas Sherson, Richard Heusdens and W. Bastiaan Kleijn**, *Derivation and Analysis of the Primal-Dual Method of Multipliers Based on Monotone Operator Theory*,

IEEE Transactions on Signal and Information Processing Over Networks, Accepted for Publication October 2018.

2. **Thomas Sherson, Richard Heusdens and W. Bastiaan Kleijn**, *On the Distributed Method of Multipliers for Separable Convex Optimization Problems*, IEEE Transactions on Signal and Information Processing Over Networks, Accepted for Publication February 2019.

3. **Thomas Sherson, Richard Heusdens and W. Bastiaan Kleijn**, *On the Effect of Network Topology On the Primal Dual Method Of Multipliers*, Submitted to IEEE Transactions on Signal and Information Processing Over Networks, in submission.

4. **Andreas Koutrouvelis, Thomas Sherson, Richard Heusdens and Richard Hendriks**, *A Low-Cost Robust Distributed Linearly Constrained Beamformer for Wireless Acoustic Sensor Networks with Arbitrary Topology*, IEEE/ACM Transactions on Audio, Speech and Language Processing, vol. 26, no. 8, 2018.

LIST OF CONFERENCE PAPERS

1. **Thomas Sherson, Richard Heusdens and W. Bastiaan Kleijn** ,*A Distributed Algorithm for Robust LCMV Beamforming*, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2016.

2. **Thomas Sherson, Richard Heusdens and W. Bastiaan Kleijn**, *On the duality of Globally Constrained Separable Problems and its Applciation to Distributed Signal Processing*, European Signal Processing Conference (EUSIPCO), 2016.

3. **Daan Schellekens, Thomas Sherson and Richard Heusdens**, *Quantization Effects in PDMM: A First Study for Synchronous Distributed Averaging*, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017.

4. **Jake Jonkman, Thomas Sherson and Richard Heusdens**, *Quantization Effects in Distributed Optimization*, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018.

2

MONOTONE OPERATOR THEORY AND CONVEX OPTIMIZATION

"If I have seen further, it is only by standing on the shoulders of giants."

Isaac Newton

2.1. INTRODUCTION

While this thesis focuses on the field of distributed optimization, in essence it boils down to the treatment of convex optimization via the lens of monotone operator theory. Such a perspective is useful as while the world of convex optimization is diverse and varied in nature, many first order convex solvers can be interpreted from the aforementioned monotone operator perspective. The importance of this link is that such theory can be used as a common mathematical basis in the design and analysis of different solvers. This chapter therefore serves as an introduction to this perspective. In Section 2.2 we provide a short overview of monotone operator theory and introduce appropriate definitions and theory to analyze convex problems. In Sections 2.3, 2.4 and 2.5 we then demonstrate how this theory can be used to derive and analyze a number of different convex solvers for both unconstrained and constrained optimization problems. Finally, in Section 2.6 we outline the important features of distributed optimization solvers and rederive an existing solver that meets these criteria via monotone operator theory. For the interested reader, a complete treatment of this topic can be in [34].

2.2. EUCLIDEAN SPACES SPACES AND RELATIONAL MAPPINGS

To start our discussion on monotone operator theory we begin with the notion of relational mappings. In particular, consider an N dimensional Euclidean space denoted by $\mathbb{R}^N$. Unless otherwise stated, the inner product $\langle \bullet, \bullet \rangle$ and associated norm $\|\bullet\|$ are used to denote the standard Euclidean inner product and norm respectively unless otherwise stated. Monotone operator theory can also be applied in general Hilbert spaces with similar definitions holding in those contexts. For simplicity however, we have chosen not to treat these here and to instead focus on relevant content to support the remainder of this thesis.

Let $\mathscr{X}, \mathscr{Y} \subseteq \mathbb{R}^N$ be two subsets of the Euclidean space in question. A relation or *operator* $\mathbf{T} : \mathscr{X} \to \mathscr{Y}$ describes a mapping between points in these subspaces. We can classify this mapping via the associated graph of $\mathbf{T}$ which is defined as follows.

Definition 2.2.1. *Graph of an Operator: Given a point to set operator* $\mathbf{T} : \mathscr{X} \to \mathscr{Y}$ *the graph of this operator is given by*

$$\operatorname{gra}(\mathbf{T}) = \left\{ (\mathbf{x}, \mathbf{y}) \in \mathscr{X} \times \mathscr{Y} \mid \mathbf{y} \in \mathbf{T}(\mathbf{x}) \right\}.$$

The vectors $\mathbf{x} \in \mathscr{X}$ denote those points in the domain of $\mathbf{T}$ while $\mathbf{y} \in \mathscr{Y}$ are those points in the codomain for a given $\mathbf{x}$. Note that here we allow $\mathbf{T}$ to be a general point to set mapping such that each output vector $\mathbf{y}$ need not be unique for any given $\mathbf{x}$. Similarly, there may be input vectors $\mathbf{x}$ for which the output set may be empty, i.e., $\mathbf{T}(\mathbf{x}) = \emptyset$. In the specific case that $\mathbf{y}$ is unique then the operator is referred to as single valued. All linear operators for instance are single valued operators. An additional and simple example of a single valued mapping is the derivative of a differentiable function f. In particular, if a function $f : \mathscr{X} \to \mathbb{R}$ is differentiable over its domain then

$$\forall \mathbf{x} \in \mathscr{X} : \exists \mathbf{y} \mid \nabla f(\mathbf{x}) = \mathbf{y}$$

where ∇f denotes the gradient of f.

Similarly, sub-differentiable functions offer a simple non-trivial example of point to set operators. In particular, if a function $f : \mathscr{X} \to \mathbb{R}$ is subdifferentiable then

$$\forall \mathbf{x} \in \mathscr{X} : \exists \mathbf{Y} = \left\{ \mathbf{y} \mid \mathbf{y} \in \partial f(\mathbf{x}) \right\},$$

where ∂f denotes the subgradient operator of the function f.

For general relations we can also define the notion of an operational inverse by considering the graph of an operator. Specifically, we define the operational inverse as follows.

Definition 2.2.2. *Operational Inverse: Given a point to set operator $\mathbf{T} : \mathscr{X} \to \mathscr{Y}$ its operational inverse is defined in terms of its graph so that*

$$\text{gra}\left(\mathbf{T}^{-1}\right) = \left\{ (\mathbf{y}, \mathbf{x}) \in \mathscr{Y} \times \mathscr{X} \mid \mathbf{y} \in \mathbf{T}(\mathbf{x}) \right\},$$

where we can note that we have inverted the order of the elements of the tuples to match those vectors from the domain and codomain of the inverse operator.

In the later portion of this chapter we make use of this notion of an operational inverse in the construction of a number of different first order convex solvers.

2.2.1. MONOTONE OPERATORS AND CONVEXITY

We now move our attention to a family of operators relevant to this thesis. Specifically, we will consider the set of *monotone* operators. As we will see in the coming section, this family of operators is particularly interesting in our context due to its link with convex optimization. Notably, the properties we develop here can be used to analyze a wide range of convex solvers. These basic properties therefore provide a foundation for the remaining analysis in this thesis.

An operator, be it single valued or a more general point to set mapping, is monotone if it satisfies the following condition.

Definition 2.2.3. *Monotonicity: The operator $\mathbf{T} : \mathscr{X} \to \mathscr{Y}$ is monotone if for all $(\mathbf{x}_1, \mathbf{y}_1)$, $(\mathbf{x}_2, \mathbf{y}_2) \in \text{gra}(\mathbf{T})$*

$$\left\langle \mathbf{y}_1 - \mathbf{y}_2, \mathbf{x}_1 - \mathbf{x}_2 \right\rangle \geq 0.$$

Here $\langle \bullet, \bullet \rangle$ denotes the inner product associated with the Euclidean space in question. In a one dimensional case, monotonicity essentially means that an operator preserves the directions between points in the domain and codomain. An immediate biproduct of the definition of monotone operators is that their operational inverses are also monotone. This can easily be seen by observing that any tuple $(\mathbf{x}, \mathbf{y}) \in \text{gra}(\mathbf{T})$ exactly corresponds to a tuple $(\mathbf{y}, \mathbf{x}) \in \text{gra}(\mathbf{T}^{-1})$.

A subtle but important stricter class of operators are those which are maximally monotone.

Definition 2.2.4. *Maximal Monotonicity: An operator $\mathbf{T} : \mathscr{X} \to \mathscr{Y}$ is maximal monotone if it is monotone and furthermore if there does not exist a monotone $\tilde{\mathbf{T}} : \tilde{\mathscr{X}} \to \tilde{\mathscr{Y}} \mid \mathscr{X} \subset \tilde{\mathscr{X}}, \mathscr{Y} \subset \tilde{\mathscr{Y}}$.*

In words, a monotone operator $\mathbf{T}$ is maximal if no tuple $(\mathbf{x},\mathbf{y})$ can be added to its graph without the new operator no longer being monotone. The importance of this class of operators is not explored in detail here but this property is often essential in proving convergence guarantees for convex solvers. Additionally, as with monotonicity, if an operator is maximally monotone so is its inverse.

As alluded to in the introduction to this chapter, the motivation for considering monotone operators in this thesis is their inherent link with convex optimization. In the following we extrapolate on this point by demonstrating the link between the subdifferentials of convex functions and maximal monotone operators. Consider a subdifferentiable convex function $f : \mathscr{X} \to \mathbb{R}$ which is closed, convex and proper (CCP). For shorthand, we will denote the family of functions by the set Γ_0. From the first order condition of convexity, given two vectors $\mathbf{x}_1$, $\mathbf{x}_2 \in \mathscr{X}$ we know that

$$f(\mathbf{x}_2) \geq f(\mathbf{x}_1) + \mathbf{y}_1^T (\mathbf{x}_2 - \mathbf{x}_1),$$

where $\mathbf{g}_1 \in \partial f(\mathbf{x}_1)$. Similarly, by defining $\mathbf{y}_2 \in \partial f(\mathbf{x}_2)$ we have that

$$f(\mathbf{x}_1) \geq f(\mathbf{x}_2) + \mathbf{y}_2^T (\mathbf{x}_1 - \mathbf{x}_2).$$

By summing these two inequalities and rearranging, we find that

$$\langle \mathbf{y}_1 - \mathbf{y}_2, \mathbf{x}_1 - \mathbf{x}_2 \rangle \geq 0.$$

As this must hold for all $\mathbf{y}_1$, $\mathbf{y}_2$, it follows that the subdifferentials of convex functions are monotone operators. The converse, of course, is not necessarily true as there are monotone operators which are not the subdifferentials of convex functions. In the context of this thesis, the fact that the forward relation holds is a crucial point. Additionally, under the assumption that $f \in \Gamma_0$ such operators are also maximal monotone. A proof of this point can be found in [34] for those interested.

Considering the fact that the operational inverse of a maximal monotone operator is maximal monotone, it follows that $(\partial f)^{-1}$ is also monotone. Interestingly, in [34] it was shown that given a function $f \in \Gamma_0$, that

$$\partial f^* = \left(\partial f\right)^{-1}, \; f^* = \sup_{\mathbf{x}} \left(\mathbf{y}^T \mathbf{x} - f(\mathbf{x})\right), \tag{2.1}$$

where f^* denotes the Fenchel conjugate of f. In this way the subdifferentials of conjugate functions are also maximally monotone. This in turn means that the conjugate functions f^* in this case are also CCP.

We can also draw a number of interesting links between basic properties of a given function f and associated properties of monotone operators. For instance, we already saw above that when a function is differentiable then its derivative operator is single valued. Similarly, in the case of strictly convex functions we can imply something stronger about the resulting monotone subdifferential operator. Specifically, strict convexity is defined as follows

Definition 2.2.5. *Strict Convexity: A function f is strictly convex with if for all $\theta \in [0,1], \mathbf{x}_1 \in dom\left(f\right), \mathbf{x}_2 \in dom\left(f\right), \mathbf{x}_1 \neq \mathbf{x}_2, \mathbf{y}_2 \in \partial f\left(\mathbf{x}_2\right)$*

$$f(\mathbf{x}_1) > f(\mathbf{x}_2) + \langle \mathbf{y}_2, \mathbf{x}_1 - \mathbf{x}_2 \rangle.$$

Following the same approach as we did for linking convexity and monotonicity, it follows that for such functions we can define the notion of strict monotonicity which is given below.

Definition 2.2.6. *Strictly Monotone: The operator ∂f is strictly monotone if for all $\mathbf{x}_1 \in dom(f), \mathbf{x}_2 \in dom(f)$, $\mathbf{x}_1 \neq \mathbf{x}_2$, $\mathbf{y}_1 \in \partial f(\mathbf{x}_1)$, $\mathbf{y}_2 \in \partial f(\mathbf{x}_2)$,*

$$\langle \mathbf{y}_1 - \mathbf{y}_2, \mathbf{x}_1 - \mathbf{x}_2 \rangle > 0.$$

In the case of a general strictly monotone operator $\mathbf{T}$, a byproduct of this definition is that $\mathbf{T}^{-1}$ must be single valued as no two points in the domain of $\mathbf{T}$ can map to the same point in the codomain.

Combining the above results, this leads to the fact that differentiability and strict convexity are in fact dual properties of one another with regards to Fenchel conjugation. In other words, if f is differentiable so that ∂f is single valued, then ∂f^* is strictly monotone such that f^* is strictly convex and visa versa. This duality property proves to be extremely useful when deriving solvers and in particular will be exploited when we are considering the case of solving constrained optimization problems.

2.2.2. STRONGER FUNCTIONAL PROPERTIES

Having formulated the link between convexity and monotonicity, we can leverage results from the field of monotone operator theory to both analyze existing convex solvers and devise new ones in turn. To assist in this process, in the following we introduce a number of additional statements relating functional assumptions from convex optimization with their equivalent monotone operator theory counterparts.

The first additional property we will consider is that of strong convexity which strengthens the strict convexity assumption made previously. This is defined as follows.

Definition 2.2.7. *Strong Convexity: A function f is μ-strongly convex with $\mu > 0$ if for all $\mathbf{x}_1 \in dom(f), \mathbf{x}_2 \in dom(f), \mathbf{y}_2 \in \partial f(\mathbf{x}_2)$,*

$$f(\mathbf{x}_1) \geq f(\mathbf{x}_2) + \langle \mathbf{y}_2, \mathbf{x}_1 - \mathbf{x}_2 \rangle + \frac{\mu}{2} \|\mathbf{x}_1 - \mathbf{x}_2\|^2.$$

This implies that, $f - \frac{\mu}{2}\|\bullet\|^2$ is convex.

If f is μ-strongly convex, ∂f is μ-strongly monotone which is defines as follows.

Definition 2.2.8. *Strongly Monotone: The operator ∂f is μ-strongly monotone with $\mu > 0$, if for all $\mathbf{x}_1 \in dom(f), \mathbf{x}_2 \in dom(f), \mathbf{y}_1 \in \partial f(\mathbf{x}_1), \mathbf{y}_2 \in \partial f(\mathbf{x}_2)$,*

$$\langle \mathbf{y}_1 - \mathbf{y}_2, \mathbf{x}_1 - \mathbf{x}_2 \rangle \geq \mu \|\mathbf{x}_1 - \mathbf{x}_2\|^2$$

The next major property strengthens the notions of differentiability by also enforcing that the function itself is smooth. Specifically, the smoothness of a function is defined in the following.

Definition 2.2.9. *Smoothness: A convex function f is β-smooth with $\beta > 0$ if it is both differentiable and for all $\mathbf{x}_1 \in dom(f), \mathbf{x}_2 \in dom(f)$,*

$$f(\mathbf{x}_1) \leq f(\mathbf{x}_2) + \langle \nabla f(\mathbf{x}_2), \mathbf{x}_1 - \mathbf{x}_2 \rangle + \frac{\beta}{2} \|\mathbf{x}_1 - \mathbf{x}_2\|^2.$$

This implies that, $\frac{\beta}{2}\|\bullet\|^2 - f$ is convex.

If f is β-smooth, ∇f is $\frac{1}{\beta}$-cocoercive which is defined as follows.

Definition 2.2.10. *Cocoercive: The monotone operator ∇f is $\frac{1}{\beta}$-cocoercive with $\beta > 0$ if for all $\mathbf{x}_1 \in dom(f), \mathbf{x}_2 \in dom(f)$,*

$$\langle \nabla f(\mathbf{x}_1) - \nabla f(\mathbf{x}_2), \mathbf{x}_1 - \mathbf{x}_2 \rangle \geq \frac{1}{\beta} \|\nabla f(\mathbf{x}_1) - \nabla f(\mathbf{x}_2)\|^2.$$

Furthermore, if f is β-smooth, ∂f is β-Lipschitz continuous, again defined below.

Definition 2.2.11. *Lipschitz Continuous: The operator ∇f is β-Lipschitz if for all $\mathbf{x}_1 \in dom(f), \mathbf{x}_2 \in dom(f)$,*

$$\|\nabla f(\mathbf{x}_1) - \nabla f(\mathbf{x}_2)\| \leq \beta \|\mathbf{x}_1 - \mathbf{x}_2\|$$

The notion of Lipschitz continuity can be used to define the notions of nonexpansiveness and contractiveness.

Definition 2.2.12. *Nonexpansive and Contractive Operators: A β-Lipschitz operator ∇f is nonexpansive if $\beta = 1$ and contractive if $\beta < 1$.*

The nonexpansiveness of an operator plays a central role in the design of numerous convex solvers. Specifically, if an operator is nonexpansive, any two points mapped under said operator are at least as close in the codomain as they were in the domain. As we will show in the coming sections, this point lies at the heart of many convergence proofs.

As in the case of differentiability and strict convexity, these stronger properties, namely smoothness and strong convexity are also duals of one another under Lagrangian duality. Specifically, if a function $f \in \Gamma_0$ is β-smooth then its Fenchel conjugate f^* is $\frac{1}{\beta}$-strongly convex. Similarly, if a function $f \in \Gamma_0$ is μ-strongly convex then its Fenchel conjugate f^* is $\frac{1}{\mu}$-smooth in turn.

2.2.3. MANIPULATIONS OF OPERATORS

The final piece of the puzzle in forming many of the solvers introduced in the remainder of this chapter is to demonstrate some manipulated forms of maximal monotone operators. These manipulations include some basic operations which preserve monotonicity as well as others which allow us to form nonexpansive operators from maximal monotone operators.

SUMS OF MONOTONE OPERATORS

A straighforward but important property of monotone operators is that their monotonicity is preserved under summation. Specifically, given two monotone operators $\mathbf{T}_1 : \mathscr{X}_1 \rightarrow \mathscr{Y}_1$ and $\mathbf{T}_2 : \mathscr{X}_2 \rightarrow \mathscr{Y}_2$ their summation $\mathbf{T}_1 + \mathbf{T}_2$ is also monotone if $\mathscr{X}_1 \cap \mathscr{X}_2 \neq \emptyset$, i.e., their domains share a common point.

To prove that summations preserve monotonicity, consider the following. Let $\mathbf{x}_a, \mathbf{x}_b \in \mathscr{X}_1 \cap \mathscr{X}_2 \neq \emptyset$. For any such point define $\mathbf{y}_{a,1} \in \mathbf{T}_1(\mathbf{x}_a)$, $\mathbf{y}_{a,2} \in \mathbf{T}_2(\mathbf{x}_a)$, $\mathbf{y}_{b,1} \in \mathbf{T}_1(\mathbf{x}_b)$, $\mathbf{y}_{b,2} \in \mathbf{T}_2(\mathbf{x}_b)$ such that $\mathbf{y}_a = \mathbf{y}_{a,1} + \mathbf{y}_{a,2} \in \mathbf{T}_1(\mathbf{x}_a) + \mathbf{T}_2(\mathbf{x}_a)$, $\mathbf{y}_b = \mathbf{y}_{b,1} + \mathbf{y}_{b,2} \in \mathbf{T}_1(\mathbf{x}_b) + \mathbf{T}_2(\mathbf{x}_b)$.

From the definition of monotonicity, it follows that

$$\begin{aligned}\langle \mathbf{y}_a - \mathbf{y}_b, \mathbf{x}_a - \mathbf{x}_b \rangle &= \langle \mathbf{y}_{a,1} + \mathbf{y}_{a,2} - \mathbf{y}_{b,1} - \mathbf{y}_{b,2}, \mathbf{x}_a - \mathbf{x}_b \rangle \\ &= \langle \mathbf{y}_{a,1} - \mathbf{y}_{b,1}, \mathbf{x}_a - \mathbf{x}_b \rangle + \langle \mathbf{y}_{a,2} - \mathbf{y}_{b,2}, \mathbf{x}_a - \mathbf{x}_b \rangle \geq 0,\end{aligned}$$

where the final inequality stems from the monotonicity of $\mathbf{T}_1$ and $\mathbf{T}_2$.

COMPOSITIONS OF MONOTONE OPERATORS AND LINEAR OPERATORS

An equally straighforward but important property is that of compositions of monotone and linear operators. In particular, given a monotone operator $\mathbf{T} : \mathscr{Y} \rightarrow \mathscr{Y}$ and a linear mapping $\mathbf{A} : \mathscr{X} \rightarrow \mathscr{Y}$, the composition $\mathbf{A}^T \circ \mathbf{T} \circ \mathbf{A} : \mathscr{X} \rightarrow \mathscr{X}$ is monotone. The notation $\circ$ is used to denote the composition of two operators. For instance, given the operators $\mathbf{T}_1 : \mathscr{X} \rightarrow \mathscr{Y}$ and $\mathbf{T}_2 : \mathscr{Y} \rightarrow \mathscr{Z}$ their composition $\mathbf{T}_2 \circ \mathbf{T}_1$ implies $\forall\, (\mathbf{x}, \mathbf{z}) \in \text{gra}\,(\mathbf{T}_2 \circ \mathbf{T}_1)\,, \exists \mathbf{y} |\, (\mathbf{x}, \mathbf{y}) \in \text{gra}\,(\mathbf{T}_1)\,, (\mathbf{y}, \mathbf{z}) \in \text{gra}\,(\mathbf{T}_2)$

To prove that the composition of monotone and linear operators $\mathbf{A}^T \circ \mathbf{T} \circ \mathbf{A}$ is monotone, consider the vectors $\mathbf{x}_1, \mathbf{x}_2 \in \mathscr{X}$, $\mathbf{y}_1 \in \mathbf{T} \circ \mathbf{A}(\mathbf{x}_1), \mathbf{y}_2 \in \mathbf{T} \circ \mathbf{A}(\mathbf{x}_2)$. It follows that

$$\langle \mathbf{A}^T (\mathbf{y}_1 - \mathbf{y}_2), \mathbf{x}_1 - \mathbf{x}_2 \rangle = \langle \mathbf{y}_1 - \mathbf{y}_2, \mathbf{A}(\mathbf{x}_1 - \mathbf{x}_2) \rangle \geq 0.$$

The final inequality follows from the linearity of the operator $\mathbf{A}$, i.e., that $\mathbf{A}^T$ is the conjugate of $\mathbf{A}$ in this case, and that $\mathbf{y}_1 \in \mathbf{T} \circ \mathbf{A}(\mathbf{x}_1), \mathbf{y}_2 \in \mathbf{T} \circ \mathbf{A}(\mathbf{x}_2)$.

Assuming that $\mathbf{T}$ is the subdifferential of some function f, if we assume that f is β-Lipschitz continuous, it follows that $\mathbf{A}^T \nabla f \mathbf{A}$ is $\beta \sigma_{\max}^2(\mathbf{A})$ Lipschitz continuous where $\sigma_{\max}(\mathbf{A})$ denotes the largest singular value of $\mathbf{A}$. To observe this, we can use the definition of Lipschitz continuity to note that

$$\begin{aligned}\left\| \mathbf{A}^T (\nabla f(\mathbf{A}\mathbf{x}_1) - \nabla f(\mathbf{A}\mathbf{x}_2)) \right\| &\leq \sigma_{\max}(\mathbf{A}) \left\| \nabla f(\mathbf{A}\mathbf{x}_1) - \nabla f(\mathbf{A}\mathbf{x}_2) \right\| \\ &\leq \beta \sigma_{\max}(\mathbf{A}) \left\| \mathbf{A}(\mathbf{x}_1 - \mathbf{x}_2) \right\| \leq \beta \sigma_{\max}^2(\mathbf{A}) \left\| \mathbf{x}_1 - \mathbf{x}_2 \right\|,\end{aligned}$$

where the penultimate line uses the Lipschitz continuity of f.

Similarly, if f is μ-strongly convex and $\mathbf{A}$ has full row rank then $\mathbf{A}^T \nabla f \mathbf{A}$ is $\mu \sigma_{\min}^2(\mathbf{A})$ strongly convex where $\sigma_{\min}^2(\mathbf{A})$ denotes the smallest singular value of $\mathbf{A}$. This can be observed by using the definition of strong convexity by noting that $\forall \mathbf{x}_1, \mathbf{x}_2 \in \text{dom}\,(f), \mathbf{y}_1 \in \partial f(\mathbf{A}\mathbf{x}_1), \mathbf{y}_2 \in \nabla f(\mathbf{A}\mathbf{x}_2)$

$$\begin{aligned}\langle \mathbf{A}^T (\mathbf{y}_1 - \mathbf{y}_2), \mathbf{x}_1 - \mathbf{x}_2 \rangle = \langle \mathbf{y}_1 - \mathbf{y}_2, \mathbf{A}(\mathbf{x}_1 - \mathbf{x}_2) \rangle &\geq \mu \left\| \mathbf{A}(\mathbf{x}_1 - \mathbf{x}_2) \right\|^2 \\ &\geq \mu \sigma_{\min}^2(\mathbf{A}) \left\| \mathbf{x}_1 - \mathbf{x}_2 \right\|^2.\end{aligned}$$

where we have again made use of the linearity of $\mathbf{A}$.

RESOLVENT OF MONOTONE OPERATORS

A key manipulation of monotone operators and one which we shall use extensively throughout this thesis is the so called *resolvent*. Such operators show up in a wide number of solvers for both unconstrained and constrained optimization as we will see in Sections 2.3 and 2.4. The resolvent operator is defined as follows.

Definition 2.2.13. *Resolvent: Given an operator* $\mathbf{T} : \mathscr{X} \to \mathscr{X}$ *and a positive scalar* $\rho > 0$, *the associated resolvent operator is given by*

$$\mathbf{J}_{\mathbf{T},\rho} = \left(\mathbf{I} + \rho\mathbf{T}\right)^{-1}$$

In the case that $\mathbf{T}$ is monotone, we can note that the resolvent operator is nonexpansive. This follows directly from the fact that the operator $\mathbf{I} + \rho\mathbf{T}$ is at least 1-strongly monotone such that its inverse is 1 Lipschitz continuous. The strong monotonicity of $\mathbf{I} + \rho\mathbf{T}$ also ensures that the resolvent operator is single valued. Additionally, in the case that $\mathbf{T}$ is maximal monotone the domain of the resolvent operator is the entire Euclidean space $\mathbb{R}^N$. A proof of this property can be found in [34][Theorem 21.1].

The nonexpansiveness of the resolvent of monotone operators is an important property as it means that the operator is stable, i.e. that it does not increase the distance between points mapped under it. In the case of optimization solvers, this type of property is one way of showing that an algorithm will not diverge away from a good solution. However, nonexpansiveness by itself is typically not strong enough to guarantee that an algorithm will converge.

In the case that $\mathbf{T}$ is monotone, we can strengthen the nonexpansiveness of $\mathbf{J}_{\mathbf{T},\rho}$ by showing that it is firmly nonexpansive which is defined as follows.

Definition 2.2.14. *Firmly Nonexpansive Operators: An operator* $\mathbf{T} : \mathscr{X} \to \mathscr{Y}$ *is firmly nonexpansive if* $\forall \mathbf{x}_1, \mathbf{x}_2 \in \mathscr{X}$, $\mathbf{y}_1 \in \mathbf{T}(\mathbf{x}_1)$, $\mathbf{y}_2 \in \mathbf{T}(\mathbf{x}_2)$

$$\left\|\mathbf{y}_1 - \mathbf{y}_2\right\|^2 + \left\|\mathbf{x}_1 - \mathbf{y}_1 - (\mathbf{x}_2 - \mathbf{y}_2)\right\|^2 \leq \|\mathbf{x}_1 - \mathbf{x}_2\|^2.$$

From [34, Corollary 4.5] if we can also extend this definition by noting that the following statements are equivalent.

- The operator $\mathbf{T}$ is firmly nonexpansive.
- The operator $\mathbf{I} - \mathbf{T}$ is firmly nonexpansive.
- The operator $2\mathbf{T} - \mathbf{I}$ is nonexpansive.
- $\left\|\mathbf{y}_1 - \mathbf{y}_2\right\|^2 \leq \left\langle \mathbf{y}_1 - \mathbf{y}_2, \mathbf{x}_1 - \mathbf{x}_2 \right\rangle$.

To prove that the resolvent of a monotone operator is firmly non-expansive, consider a monotone operator $\mathbf{T} : \mathscr{X} \to \mathscr{Y}$ and the vectors $\mathbf{x}_1, \mathbf{x}_2 \in \mathscr{X}$, $\mathbf{y}_1 = \mathbf{J}_{\mathbf{T},\rho}(\mathbf{x}_1)$, $\mathbf{y}_2 = \mathbf{J}_{\mathbf{T},\rho}(\mathbf{x}_2)$. By definition of the resolvent operator and the operational inverse, we have that

$$\mathbf{x}_1 \in \mathbf{y}_1 + \rho\mathbf{T}(\mathbf{y}_1),\ \mathbf{x}_2 \in \mathbf{y}_2 + \rho\mathbf{T}(\mathbf{y}_2).$$

Taking the difference of both inclusions we find that

$$\mathbf{x}_1 - \mathbf{x}_2 \in \mathbf{y}_1 + \rho\mathbf{T}(\mathbf{y}_1) - \left(\mathbf{y}_2 + \rho\mathbf{T}(\mathbf{y}_2)\right).$$

By then taking the inner product of both sides with $\mathbf{y}_1 - \mathbf{y}_2$ it follows that

$$\langle \mathbf{x}_1 - \mathbf{x}_2, \mathbf{y}_1 - \mathbf{y}_2 \rangle \in \|\mathbf{y}_1 - \mathbf{y}_2\|^2 + \rho \langle \mathbf{y}_1 - \mathbf{y}_2, \mathbf{T}(\mathbf{y}_1) - \mathbf{T}(\mathbf{y}_2) \rangle.$$

Using the monotonicity of $\mathbf{T}$ and rearranging we find that

$$\|\mathbf{y}_1 - \mathbf{y}_2\|^2 \leq \langle \mathbf{y}_1 - \mathbf{y}_2, \mathbf{x}_1 - \mathbf{x}_2 \rangle,$$

or equivalently, that

$$\|\mathbf{J}_{\mathbf{T},\rho}(\mathbf{x}_1) - \mathbf{J}_{\mathbf{T},\rho}(\mathbf{x}_2)\|^2 \leq \langle \mathbf{J}_{\mathbf{T},\rho}(\mathbf{x}_1) - \mathbf{J}_{\mathbf{T},\rho}(\mathbf{x}_2), \mathbf{x}_1 - \mathbf{x}_2 \rangle.$$

Ultimately, from the results of [34, Corollary 4.5] this implies that the resolvent is firmly nonexpansive as desired.

REFLECTED RESOLVENT OF MONOTONE OPERATORS

Another basic modification based on the resolvent operator is that of the reflected resolvent or Cayley operator defined as follows

Definition 2.2.15. *Reflected Resolvent (Cayley): Given an operator* $\mathbf{T}$ *and a positive scalar* $\rho > 0$*, the reflected resolvent operator is given by*

$$\mathbf{R}_{\mathbf{T},\rho} = 2\mathbf{J}_{\mathbf{T},\rho} - \mathbf{I}.$$

As in the case of the resolvent operator, in the case that $\mathbf{T}$ is maximal monotone, the domain of $\mathbf{R}_{\mathbf{T},\rho}$ is the entire Euclidean space $\mathbb{R}^N$. In the case of monotone operators, the reflected resolvent can be shown to be nonexpansive. This can be proved in a manner similar to that of the resolvent. Consider a monotone operator $\mathbf{T} : \mathscr{X} \to \mathscr{Y}$ and the vectors $\mathbf{x}_1, \mathbf{x}_2 \in \mathscr{X}$, $\mathbf{y}_1 \in \mathbf{R}_{\mathbf{T},\rho}(\mathbf{x}_1)$, $\mathbf{y}_2 \in \mathbf{R}_{\mathbf{T},\rho}(\mathbf{x}_2)$. By the definitions of the reflected resolvent and resolvent we have that

$$\mathbf{y}_1 \in \left(2\left(\mathbf{I} + \rho\mathbf{T}\right)^{-1} - \mathbf{I}\right)(\mathbf{x}_1),\ \mathbf{y}_2 \in \left(2\left(\mathbf{I} + \rho\mathbf{T}\right)^{-1} - \mathbf{I}\right)(\mathbf{x}_2).$$

Rearranging these inclusions and using the definition of the operational inverse, it follows that

$$\mathbf{x}_1 \in \frac{\mathbf{y}_1 + \mathbf{x}_1}{2} + \rho\mathbf{T}\left(\frac{\mathbf{y}_1 + \mathbf{x}_1}{2}\right),\ \mathbf{x}_2 \in \frac{\mathbf{y}_2 + \mathbf{x}_2}{2} + \rho\mathbf{T}\left(\frac{\mathbf{y}_2 + \mathbf{x}_2}{2}\right).$$

By then taking the difference of these inclusions we find that

$$\mathbf{x}_1 - \mathbf{x}_2 \in \frac{\mathbf{y}_1 + \mathbf{x}_1}{2} - \frac{\mathbf{y}_1 + \mathbf{x}_1}{2} + \rho\mathbf{T}\left(\frac{\mathbf{y}_1 + \mathbf{x}_1}{2}\right) - \rho\mathbf{T}\left(\frac{\mathbf{y}_1 + \mathbf{x}_1}{2}\right).$$

Taking the inner product of both sides with respect to $\frac{\mathbf{y}_1 + \mathbf{x}_1}{2} - \frac{\mathbf{y}_1 + \mathbf{x}_1}{2}$ and using the monotonicity of $\mathbf{T}$, it follows that

$$\left\langle \mathbf{x}_1 - \mathbf{x}_2, \frac{\mathbf{y}_1 + \mathbf{x}_1}{2} - \frac{\mathbf{y}_1 + \mathbf{x}_1}{2} \right\rangle \geq \left\| \frac{\mathbf{y}_1 + \mathbf{x}_1}{2} - \frac{\mathbf{y}_1 + \mathbf{x}_1}{2} \right\|^2.$$

Expanding both sides, we can form the inequality

$$\frac{1}{2}\|\mathbf{x}_1 - \mathbf{x}_2\|^2 + \frac{1}{2}\langle \mathbf{x}_1 - \mathbf{x}_2, \mathbf{y}_1 - \mathbf{y}_1 \rangle \geq \frac{1}{4}\|\mathbf{x}_1 - \mathbf{x}_2\|^2 + \frac{1}{4}\|\mathbf{y}_1 - \mathbf{y}_2\|^2 + \frac{1}{2}\langle \mathbf{x}_1 - \mathbf{x}_2, \mathbf{y}_1 - \mathbf{y}_1 \rangle.$$

Cancelling common terms therefore demonstrates that $\|\mathbf{x}_1-\mathbf{x}_2\|^2 \geq \|\mathbf{y}_1-\mathbf{y}_2\|^2$ and thus that $\mathbf{R}_{\mathbf{T},\rho}$ is nonexpansive as desired. Alternatively, given the firm nonexpansiveness of $\mathbf{J}_{\mathbf{T},\rho}$, the result follows immediately from the equivalent definitions of firmly nonexpansive operators.

AVERAGED OPERATORS

The major difference in properties between the resolvent and reflected resolvent operators is that the prior was firmly nonexpansive. More generally speaking the resolvent operator is an averaged version of the reflected resolvent. An averaged operator is defined as follows.

Definition 2.2.16. *Averaged Operator: An operator $\mathbf{T}:\mathscr{X}\to\mathscr{X}$ is an averaged operator if there exists a nonexpansive operator $\mathbf{S}:\mathscr{X}\to\mathscr{X}$, $\alpha\in(0,1)$ such that*

$$\mathbf{T}=(1-\alpha)\mathbf{I}+\alpha\mathbf{S}.$$

In the case of the resolvent $\mathbf{J}_{\mathbf{T},\rho}$ we can verify it is averaged by setting $\alpha=\frac{1}{2}$ and $\mathbf{S}=\mathbf{R}_{\mathbf{T},\rho}$. Using these definitions, it follows that

$$\frac{1}{2}\left(\mathbf{I}+\mathbf{R}_{\mathbf{T},\rho}\right)=\frac{1}{2}\left(\mathbf{I}+2\mathbf{J}_{\mathbf{T},\rho}-\mathbf{I}\right)=\mathbf{J}_{\mathbf{T},\rho},$$

as desired. More generally, we can use a similar point to demonstrate the equivalence of $\frac{1}{2}$-averaged and firmly non-expansive operators. Notably, we already know that an operator $\mathbf{T}$ is firmly nonexpansive if and only if the operator $\mathbf{S}=2\mathbf{T}-\mathbf{I}$ is nonexpansive. With simple manipulation of this equality, we find that $\mathbf{T}=\frac{1}{2}(\mathbf{I}+\mathbf{S})$ which is therefore both $\frac{1}{2}$-averaged and firmly nonexpansive thus demonstrating their equivalence.

In general, operator averaging is a useful property to strengthen the characteristics of an operator. Specifically, we can make use of an interesting norm identity given in [34, Corollary 2.15] which we include here for completeness. Given two vectors $\mathbf{x}_1,\mathbf{x}_2$,

$$\|(1-\alpha)\mathbf{x}_1+\alpha\mathbf{x}_2\|^2=(1-\alpha)\|\mathbf{x}_1\|^2+\alpha\|\mathbf{x}_2\|^2-\alpha(1-\alpha)\|\mathbf{x}_1-\mathbf{x}_2\|^2. \tag{2.2}$$

In the context of averaged operators, we can apply this property to show the following. Consider a nonexpansive operator $\mathbf{T}:\mathscr{X}\to\mathscr{X}$, a scalar $\alpha\in(0,1)$ and the set of vectors $\mathbf{x}_1,\mathbf{x}_2\in\mathscr{X}$, $\mathbf{y}_1\in\mathbf{T}(\mathbf{x}_1)$, $\mathbf{y}_2\in\mathbf{T}(\mathbf{x}_2)$, $\mathbf{z}_1=(1-\alpha)\mathbf{x}_1+\alpha\mathbf{y}_1$, $\mathbf{z}_2=(1-\alpha)\mathbf{x}_2+\alpha\mathbf{y}_2)$. It follows form (2.2) that

$$\begin{aligned}\|\mathbf{z}_1-\mathbf{z}_2\|^2&=(1-\alpha)\|\mathbf{x}_1-\mathbf{x}_2\|^2+\alpha\|\mathbf{y}_1-\mathbf{y}_2\|^2-\alpha(1-\alpha)\|\mathbf{x}_1-\mathbf{x}_2-(\mathbf{y}_1-\mathbf{y}_2)\|^2\\&\leq\|\mathbf{x}_1-\mathbf{x}_2\|^2-\alpha(1-\alpha)\|\mathbf{x}_1-\mathbf{x}_2-(\mathbf{y}_1-\mathbf{y}_2)\|^2,\end{aligned} \tag{2.3}$$

where the final line stems from the nonexpansiveness of $\mathbf{R}_{\mathbf{T},\rho}$. Unlike a nonexpansive operator, averaged operator therefore guarantee that points in the codomain are strictly closer to each other than their domain counterparts unless $\mathbf{x}_1-\mathbf{x}_2-(\mathbf{y}_1-\mathbf{y}_2)=\mathbf{0}$. As we will see in the following section, operator averaging can be an effective means of guaranteeing the convergence of an algorithm when nonexpansiveness itself is not sufficient.

2.2.4. FINDING FIXED POINTS OF NONEXPANSIVE OPERATORS

The final basic theory we require before using monotone operator theory to analyze convex optimization solvers relates to methods for finding fixed points of nonexpansive operators. In the following we introduce one such approach, that being the Banach-Picard iteration to achieve this. In particular, given a nonexpansive operator $\mathbf{T}$, the Banach-Picard iteration aims to find a vector $\mathbf{x}$ so that

$$\mathbf{x} \in \mathbf{T}(\mathbf{x}).$$

In other words, the Banach-Picard iteration wants to find a vector $\mathbf{x}^*$ so that $(\mathbf{x}^*, \mathbf{x}^*) \in \text{gra}(\mathbf{T})$. As previously mentioned, such vectors are referred to as the fixed points of an operator and the set of all such points are denoted by $\text{fix}(\mathbf{T})$ in the case of an operator $\mathbf{T}$.

For a given operator $\mathbf{T}$ and a previous guess at a fixed point vector $\mathbf{x}^{(k)}$, the Banach-Picard iteration forms a new vector $\mathbf{x}^{(k+1)}$ as

$$\mathbf{x}^{(k+1)} = \mathbf{T}\left(\mathbf{x}^{(k)}\right).$$

This updating procedure is then applied repeatedly to compute a fixed point, if such a point exists. In the case of nonexpansive operators, it can be shown that this procedure is stable, i.e. that the sequence of iterates generated, which we can denote via $\left(\mathbf{x}^{(k)}\right)_{k\in\mathbb{N}}$, does not diverge away from a given fixed point $\mathbf{x}^* \in \text{fix}(\mathbf{T})$. This point can be demonstrated by noting that

$$\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\right\|^2 = \left\|\mathbf{T}\left(\mathbf{x}^{(k)}\right) - \mathbf{T}\left(\mathbf{x}^*\right)\right\|^2 \leq \left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\|^2,$$

where the final inequality stems from the nonexpansiveness of $\mathbf{T}$.

While nonexpansiveness is sufficient to ensure that the sequence of iterates generated by the Banach-Picard iteration do not diverge from a fixed point, in general it is not sufficient to guarantee convergence. A simple example of this is the nonexpansive operator $-\mathbf{I}$. Clearly, the only fixed point of this operator is the all zeros vector yet for any other initial guess $\mathbf{x}^{(0)} \neq \mathbf{0}$, the sequence of iterates generated by the Banach-Picard iteration will not converge, instead flipping between the two states $\mathbf{x}^{(0)}$ and $-\mathbf{x}^{(0)}$. However, if we place stronger assumptions on our operator $\mathbf{T}$, i.e., that it is contractive or averaged, convergence to such a fixed point can be verified. In the case of a β-contractive operator, where $\beta < 1$, this result follows by again considering the quadratic form $\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\right\|^2$ and noting that

$$\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\right\|^2 = \left\|\mathbf{T}\left(\mathbf{x}^{(k)}\right) - \mathbf{T}\left(\mathbf{x}^*\right)\right\|^2 \leq \beta^2 \left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\|^2 \leq \beta^{2k} \left\|\mathbf{x}^{(0)} - \mathbf{x}^*\right\|^2,$$

where $\mathbf{x}^{(0)}$ denotes some initial vector used in this iterative process. It follows that as $\beta < 1$ that the distance between $\mathbf{x}^{(k)}$ and $\mathbf{x}^*$ must decrease at a geometric rate. Furthermore, this also means that $\mathbf{x}^* = \text{fix}(\mathbf{T})$ is a unique point.

In general, the operators we are interested in may not be contractive but might exhibit the weaker property of being averaged. Fortunately, we can again show that this property is sufficient for the Banach-Picard iteration to converge to a fixed point. Specifically, by considering the squared norm term $\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\right\|^2$, for an averaged operator

$\mathbf{T} = (1-\alpha)\mathbf{I} + \alpha\mathbf{S}$ it follows that

$$\begin{aligned}\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\right\|^2 &= \left\|\mathbf{T}\left(\mathbf{x}^{(k)}\right) - \mathbf{T}\left(\mathbf{x}^*\right)\right\|^2 = \left\|((1-\alpha)\mathbf{I} + \alpha\mathbf{S})\left(\mathbf{x}^{(k)}\right) - ((1-\alpha)\mathbf{I} + \alpha\mathbf{S})\left(\mathbf{x}^*\right)\right\|^2 \\ &\leq \left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\|^2 - \alpha(1-\alpha)\left\|\mathbf{S}\left(\mathbf{x}^{(k)}\right) - \mathbf{x}^{(k)}\right\|^2,\end{aligned}$$

where the final line stems from (2.3). Applying this property recursively, we find that

$$\alpha(1-\alpha)\sum_{i=0}^{k}\left\|\mathbf{S}\left(\mathbf{x}^{(i)}\right) - \mathbf{x}^{(i)}\right\|^2 \leq \left\|\mathbf{x}^{(0)} - \mathbf{x}^*\right\|^2.$$

By letting k tend to infinity, it follows that $\left\|\mathbf{S}\left(\mathbf{x}^{(k)}\right) - \mathbf{x}^{(k)}\right\|^2 \to 0$ and thus, due to the finite dimensional nature of the problems considered, that $\mathbf{S}\left(\mathbf{x}^{(k)}\right) - \mathbf{x}^{(k)} \to 0$. The notation $\bullet \to \bullet$ is used here to denote convergence. It therefore follows that the averagedness of an operator is sufficient to guarantee convergence of the Banach-Picard iterations to a fixed point of the operator $\mathbf{S}$ and by association of the operator $\mathbf{T}$.

In the following sections, we make use of Banach-Picard iterations to demonstrate the convergence characteristics of a number of different convex optimization solvers. Specifically, it can be shown that many classic solvers from within the literature can be interpreted as instances of Banach-Picard iterations applied to averaged or contractive operators, allowing us to understand their performance from their fundamental link with monotone operator theory.

2.3. UNCONSTRAINED OPTIMIZATION

With a set of basic properties and manipulations of monotone operators under our belt we are now ready to move to applying this information to the task of distributed solver design. To begin, we will consider the task of solving unconstrained convex problems. Due to their simple structure, such problems provide a good platform to demonstrate the link between monotone operator theory and first order solver design.

2.3.1. SUBGRADIENT DESCENT

Perhaps the simplest method of solving unconstrained convex optimization problems is that of subgradient descent. Given the insight into the connection between the subgradients of convex functions and monotone operator theory, this also makes subgradient descent the ideal starting point for showing how this theory can be use in the analysis of different convex solvers. With this in mind, consider a simple unconstrained optimization problem given by

$$\min_{\mathbf{x}\in\mathbb{R}^N} \quad f(\mathbf{x}), \tag{2.4}$$

where the function $f \in \Gamma_0$. Subgradient descent aims to solve this problem through the evalutation of the subgradient operator itself.

There are a number of ways in which subgradient descent can be interpreted. Perhaps the most intuitive approach stems from the perspective of convexity. As convex functions possess a unique minimum value and as the subgradient of the function will always contain a descent direction, we can simply find this minimum by moving along

any such path. Therefore, given some subdifferentiable convex function $f:\mathbb{R}^N \to \mathbb{R}$ with subgradient ∂f, and a given estimate of the optimization variables $\mathbf{x}^{(k+1)}$, mathematically we can write this procedure as

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \rho \mathbf{g}^{(k)}, \tag{2.5}$$

where the additional vector $\mathbf{g}^{(k)} \in \partial f\left(\mathbf{x}^{(k)}\right)$ while $\rho > 0$ is referred to as the step size. Equivalently we can interpret this from an operator perspective. From the first order condition of convexity, we know that for the vector $\mathbf{x}$ to be a minimizer of (2.4) it must also satisfy the monotonic inclusion

$$\mathbf{0} \in \partial f(\mathbf{x}). \tag{2.6}$$

We can therefore derive the subgradient descent approach by rephrasing (2.6) as a fixed point condition as introduced in Section 2.2.4 by noting that

$$\mathbf{0} \in \partial f(\mathbf{x}) \iff \mathbf{x} \in \left(\mathbf{I} + \rho \partial f\right)(\mathbf{x}) \iff \mathbf{x} \in \left(\mathbf{I} - \rho \partial f\right)(\mathbf{x}),$$

where the additional scalar $\rho > 0$ is the step size of the algorithm. Given this fixed point form, we can then apply classic fixed point finding techniques such as the Banach-Picard iteration to arrive at the subgradient descent algorithm given in (2.5). We can therefore think of gradient descent as a forward step type algorithm where the monotone operator ∂f is evalutated at each iteration.

While the convex interpretation of gradient descent is somewhat more intuitive, by directly noting the connection with monotone operator theory we can quickly derive sufficient conditions for algorithmic convergence. For instance, consider the case of a constant step-size algorithm where the underlying function is β-smooth. An immediate by-product of this assumption is that f must be differentiable such that $\partial f = \nabla f$. By restricting the problem class, we can show that the sequence of iterates $\left(\mathbf{x}^{(k+1)}\right)_{k\in\mathbb{N}}$ converge towards an optimal state.

To demonstrate convergence, firstly choose an $\mathbf{x}^* \in \mathbf{X}^*$ where $\mathbf{X}^*$ denotes the set of minimizers of (2.4). For each iteration k, the squared Euclidean distance between the variables $\mathbf{x}^{(k+1)}$ and this optimal point $\mathbf{x}^*$ is given by $\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\right\|^2$. Using the fact that ∇f is single valued, combined with the definition of the gradient descent algorithm, it follows that

$$\begin{aligned}\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\right\|^2 &= \left\|\mathbf{x}^{(k)} - \mathbf{x}^* - \rho\left(\nabla f\left(\mathbf{x}^{(k)}\right) - \nabla f\left(\mathbf{x}^*\right)\right)\right\|^2 \\ &= \left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\|^2 + \left\|\rho\left(\nabla f\left(\mathbf{x}^{(k)}\right) - \nabla f\left(\mathbf{x}^*\right)\right)\right\|^2 \\ &\quad - 2\left\langle \rho\left(\nabla f\left(\mathbf{x}^{(k)}\right) - \nabla f\left(\mathbf{x}^*\right)\right), \mathbf{x}^{(k)} - \mathbf{x}^*\right\rangle.\end{aligned} \tag{2.7}$$

We can now show that the sequence $\left(\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\right\|^2\right)_{k\in\mathbb{N}}$ is decreasing and furthermore that $\left(\left\|\nabla f\left(\mathbf{x}^{(k)}\right) - \nabla f\left(\mathbf{x}^*\right)\right\|^2\right)_{k\in\mathbb{N}}$ converges to zero as k tends to $+\infty$. To achieve this point we can utilize the smoothness assumption of our function f. Recall that specifically, for

the subdifferentials of convex functions, the β-smoothness of f implies the β-Lipschitz continuity and thus $\frac{1}{\beta}$-cocoercivity of ∇f. It therefore follows that

$$\left\langle \left(\nabla f\left(\mathbf{x}^{(k)}\right) - \nabla f\left(\mathbf{x}^*\right)\right), \mathbf{x}^{(k)} - \mathbf{x}^* \right\rangle \geq \frac{1}{\beta} \left\| \left(\nabla f\left(\mathbf{x}^{(k)}\right) - \nabla f\left(\mathbf{x}^*\right)\right) \right\|^2. \tag{2.8}$$

Rearranging the last two terms of (2.7) in combination with (2.8), it follows that

$$\left\| \mathbf{x}^{(k+1)} - \mathbf{x}^* \right\|^2 \leq \left\| \mathbf{x}^{(k)} - \mathbf{x}^* \right\|^2 - \rho\left(\frac{2}{\beta} - \rho\right) \left\| \nabla f\left(\mathbf{x}^{(k)}\right) - \nabla f\left(\mathbf{x}^*\right) \right\|^2,$$

and thus that restricting $\rho \in (0, \frac{2}{\beta})$ guarantees that $\left(\left\| \mathbf{x}^{(k)} - \mathbf{x}^* \right\|^2\right)_{k\in\mathbb{N}}$ forms a nonincreasing sequence and thus that $\nabla f\left(\mathbf{x}^{(k)}\right) - \nabla f\left(\mathbf{x}^*\right) \to 0$. In the case of finite dimensional problems, as f is a smooth function, the convergence of the gradient in turn ensures that $\mathbf{x}^{(k+1)} \to \mathbf{x} \in \mathbf{X}^*$.

We can strengthen the convergence result for gradient descent by making stronger assumptions on the function f. Namely, if we restrict f to be μ-strongly convex such that ∇f is μ-strongly monotone, we an rewrite (2.7) as

$$\left\| \mathbf{x}^{(k+1)} - \mathbf{x}^* \right\|^2 \leq \left(1 - 2\rho\mu + \rho^2\beta^2\right) \left\| \mathbf{x}^{(k)} - \mathbf{x}^* \right\|^2.$$

By restricting $\rho \in (0, \frac{2\mu}{\beta^2})$ it follows that the term $\left(1 - 2\rho\mu + \rho^2\beta^2\right) < 1$ such that $\left(\left\| \mathbf{x}^{(k+1)} - \mathbf{x}^* \right\|^2\right)_{k\in\mathbb{N}}$ forms a geometrically contracting sequence. As in the case of Banach-Picard iterations of contractive operators, from the strong convexity of f the vector $\mathbf{x}^*$ is unique in this case.

2.3.2. PROXIMAL POINT METHOD

While the guaranteed convergence of the gradient descent method for smooth functions is an appealing property, in practice we are often interested in solving problems based on a far broader class of functions. We are therefore interested in finding alternative methods which remove the need for a smoothness prior on our functions. One such approach to achieve this is the so called proximal point method. Unlike the gradient descent method, with its elegant interpretation via convexity, the proximal point method is slightly more mysterious. Thankfully, from an operator perspective it is both straightforward to derive and intuitive to understand.

Consider again the monotonic inclusion given in (2.6). As in the case of deriving the gradient descent algorithm, we can instead use an alternate form of operator manipulation to derive the proximal point method. Notably, due to the fact that $f \in \Gamma_0$, ∂f is maximal monotone, it follows that

$$\mathbf{0} \in \partial f(\mathbf{x}) \iff \mathbf{x} \in \left(\mathbf{I} + \rho\partial f\right)(\mathbf{x}) \iff \mathbf{x} = \left(\mathbf{I} + \rho\partial f\right)^{-1}(\mathbf{x}) = \mathbf{J}_{\partial f,\rho}(\mathbf{x}),$$

where we have utilized the definition of the operational inverse and the fact that the final operator is simply a resolvent operator and is thus single valued. Where the gradient descent algorithm could be thought of a forward step of our monotone operator, here the proximal algorithm is more reflective of a backwards type operation.

Applying the Banach-Picard iteration of this fixed point inclusion, we come to the proximal point method given by

$$\mathbf{x}^{(k+1)} = \mathbf{J}_{\partial f,\rho}\left(\mathbf{x}^{(k)}\right). \tag{2.9}$$

Recall from Definition 2.2.13 that for $\rho > 0$ the resolvent operator is firmly nonexpansive and therefore $\frac{1}{2}$-averaged. If we therefore consider the squared Euclidean error, it follows that

$$\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\right\|^2 \leq \left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\|^2 - \frac{1}{4}\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right\|^2 \ \forall k \in \mathbb{N}.$$

Applying this inequality recursively, it follows that $\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right\|^2 \to 0$ and thus that the algorithm converges towards a fixed point. Again, from the finite dimensionality of this problem this in turn guarantees that $\mathbf{x}^{(k+1)} \to \mathbf{x} \in \mathbf{X}^*$.

The major drawback of the proximal point method is in the evalutation of the resolvent operator itself. In particular, if we consider the update equation given in (2.9) we can rephrase this evalutation as a monotonic inclusion

$$\begin{aligned}\mathbf{x}^{(k+1)} = \left(\mathbf{I} + \rho\partial f\right)^{-1}\left(\mathbf{x}^{(k)}\right) &\iff \mathbf{x}^{(k)} \in \left(\mathbf{I} + \rho\partial f\right)\mathbf{x}^{(k+1)} \\ &\iff \mathbf{0} \in \partial f\left(\mathbf{x}^{(k+1)}\right) + \frac{1}{\rho}\left(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right).\end{aligned}$$

Solving this monotonic inclusion, and thus computing the iterates of the proximal point method, is equivalent to finding a minimizer of the following optimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^N} \quad f(\mathbf{x}) + \frac{1}{2\rho}\left\|\mathbf{x} - \mathbf{x}^{(k)}\right\|^2,$$

which is strongly convex such that at each iteration $\mathbf{x}^{(k+1)}$ is unique.

The proximal point method therefore recasts solving the original optimization problem as a sequence of regularized problems. If solving such regularized problems is computationally easier than solving the original optimization problem, such an approach may be appealing. However, in the general case, this casting may be cumbersome and may actually be more computationally expensive due to its iterative nature. Instead of presenting an optimal method to solve convex optimization problems, the motivation for introducing this method here is to preface the use of resolvent or proximal steps within convex solvers which often naturally present themselves in more complicated algorithms. This point is demonstrated in the remaining sections of this chapter where resolvent steps appear in all other methods presented.

A simple instance where the proximal operator can be efficiently evalutated is in solving the L1 norm minimization problem given by

$$\min_{\mathbf{x}\in\mathbb{R}^N} \quad \|\mathbf{x} - \mathbf{b}\|_1. \tag{2.10}$$

While this is a trivially easy problem to solve even without the proximal point method, in Section 2.4.1 we demonstrate an example where the efficient method of computing the resolvent operator of (2.10) is attractive in forming more complex solvers.

Considering using the proximal point method to solve (2.10), each evaluation of the resulting resolvent operator can be recast as the following convex optimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^N} \quad \|\mathbf{x}-\mathbf{b}\|_1 + \frac{1}{2\rho}\left\|\mathbf{x}-\mathbf{x}^{(k)}\right\|^2. \tag{2.11}$$

Considering the equivalent monotonic inclusion of (2.11), it follows that

$$\mathbf{0} \in \operatorname{sign}\left(\mathbf{x}^{(k+1)}-\mathbf{b}\right) + \frac{1}{\rho}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^{(k)}\right),$$

where the sign operator is an abuse of notation used to denote the elementwise sign of the input vector and corresponds to the subdifferential of the L1 norm function. Due to the separability of the sign operator over the elements of the input vector, for each $i \in \{1,2,\cdots N\}$ the scalar variable $\left[\mathbf{x}^{(k+1)}\right]_i$ is given by

$$\left[\mathbf{x}^{(k+1)}\right]_i = \begin{cases} \left[\mathbf{x}^{(k)}\right]_i + \rho & \text{if } \left[\mathbf{x}^{(k)}\right]_i + \rho < [\mathbf{b}]_i \\ \left[\mathbf{x}^{(k)}\right]_i - \rho & \text{if } \left[\mathbf{x}^{(k)}\right]_i - \rho > [\mathbf{b}]_i \\ [\mathbf{b}]_i & \text{otherwise} \end{cases}.$$

This update equation is equivalent to the soft thresholding operator and thus can be efficiently implemented without directly having to solve an equivalent convex optimization problem. In contrast, for other optimization problems, a more complicated solver may be required to compute each resolvent step.

Under the additional assumption of strong convexity, the proximal point method achieves geometric convergence and converges to the, now unique, optimal vector $\mathbf{x}^*$. This result again follows cleanly from monotone operator theory. To see this, consider again the squared Euclidean error

$$\left\|\mathbf{x}^{(k+1)}-\mathbf{x}^*\right\|^2 = \left\|\left(\mathbf{I}+\rho\partial f\right)^{-1}\mathbf{x}^{(k)} - \left(\mathbf{I}+\rho\partial f\right)^{-1}\mathbf{x}^*\right\|^2.$$

By definition, if f is μ-strongly convex, ∂f is μ-strongly monotone. The operator $\mathbf{I}+\rho\partial f$ is therefore $1+\rho\mu$ strongly convex so that $\left(\mathbf{I}+\rho\partial f\right)^{-1}$ is $\frac{1}{1+\rho\mu}$-Lipschitz continuous. It follows that

$$\begin{aligned}\left\|\mathbf{x}^{(k+1)}-\mathbf{x}^*\right\|^2 &= \left\|\left(\mathbf{I}+\rho\partial f\right)^{-1}\mathbf{x}^{(k)} - \left(\mathbf{I}+\rho\partial f\right)^{-1}\mathbf{x}^*\right\|^2 \\ &\leq \frac{1}{\left(1+\rho\mu\right)^2}\left\|\mathbf{x}^{(k)}-\mathbf{x}^*\right\|^2,\end{aligned}$$

and thus that the $\mathbf{x}$ variables converge at a geometric rate $\forall\,\rho > 0$.

2.4. OPERATOR SPLITTING

So far we have seen that through the lens of monotone operator theory, we can derive and verify the convergence of two simple methods for solving unconstrained optimization problems. However, the tradeoff between these two methods, namely the restricted range of functions for which gradient descent can be guaranteed to converge, and the

higher computational complexity of the proximal point method due to the need to evaluate the resolvent operator is discouraging. In particular we would like to be able to exploit as much structure within our functions as possible to compromise between these two points, namely to provide guaranteed convergence for a broad range of functions without incurring a dramatic increase in computational complexity.

One method for overcoming the limitations of the basic methods introduced above is the notion of operator splitting. In the context of convex optimization, operator splitting approaches are applicable to problems containing sums of convex functions and provide an efficient method to exploit the functional properties of individual functions in such sums. As an example, consider the following L1 regularized least squares or LASSO problem given by

$$\min_{\mathbf{x}\in\mathbb{R}^N} \quad \frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|^2+\theta\|\mathbf{x}\|_1. \tag{2.12}$$

Such problems have seen common usage in everything from machine vision [41, 42], to signal processing [43, 44, 45, 46] as a means of encouraging the sparsity of the solution vector $\mathbf{x}$ by penalizing the absolute sum of its elements (a convex approximation of L0 quasi-norm penalization).

The objective of (2.12) is comprised of two functions. The left hand term is the traditional least squares term which is smooth and is potentially also strongly convex with the added assumption that $\mathbf{A}$ has full column rank. In contrast the right hand term is the L1 norm term we introduced in Section 2.3.2 which is neither differentiable nor strongly convex. We would like to find methods to allow us to make use of the stronger functional assumptions of the least squares term without needing to resort to using the full proximal point method to solve this problem. As previously alluded to, the answer lies in the fact that we are trying to find the minima of a sum of convex functions.

If we consider the subdifferential of (2.12), and by noting that both functions are closed, convex and proper and share a common point within their domains, it follows that the equivalent monotonic inclusion is given by

$$\mathbf{0}\in\mathbf{A}^T\mathbf{A}\mathbf{x}-\mathbf{A}^T\mathbf{b}+\theta\,\mathrm{sign}(\mathbf{x}). \tag{2.13}$$

In the following we will demonstrate a number of splitting methods to solve such monotonic inclusions by recasting them as a more familiar fixed point inclusion. Note that the operator splitting methods presented here are based on two operator splitting schemes. While more general three operator splitting schemes have recently been proposed in the literature these are not addressed in this chapter as they were not utilized in the remainder of this thesis. For the interested reader, a number of key references on these approaches, and there applications in the context of optimization can be found in [47].

2.4.1. FORWARD-BACKWARD SPLITTING

The simplest two operator splitting approach is that of Forward-Backward splitting. As the name suggests, this approach produces an algorithm comprised of both forward operator evaluations, as were used in the gradient descent method, and backward or resolvent steps, as used in the proximal point method. Specifically, given two functions $f,g\in\Gamma_0$, where we also assume that f is differentiable, consider the monotonic inclusion $\mathbf{0}\in\nabla f(\mathbf{x})+\partial g(\mathbf{x})$. We can use Forward-Backward splitting to rephrase this as an

equivalent fixed point inclusion as follows

$$\begin{aligned}\mathbf{0} \in \nabla f(\mathbf{x}) + \partial g(\mathbf{x}) &\iff \mathbf{0} \in -\left(\mathbf{I} - \rho \nabla f\right)(\mathbf{x}) + \left(\mathbf{I} + \rho \partial g\right)(\mathbf{x}) \\ &\iff \left(\mathbf{I} - \rho \nabla f\right)(\mathbf{x}) \in \left(\mathbf{I} + \rho \partial g\right)(\mathbf{x}) \\ &\iff \mathbf{x} = \mathbf{J}_{\partial g, \rho} \circ \left(\mathbf{I} - \rho \nabla f\right)(\mathbf{x}).\end{aligned}$$

We can find such a fixed point via the Banach-Picard iteration, resulting in the Forward-Backward algorithm

$$\mathbf{x}^{(k+1)} = \mathbf{J}_{\partial g, \rho} \circ \left(\mathbf{I} - \rho \nabla f\right)\left(\mathbf{x}^{(k)}\right),$$

In the case of the L1-regularized least squares problem given in (2.13), by selecting $\nabla f(\mathbf{x}) = \mathbf{A}^T\mathbf{A}\mathbf{x} - \mathbf{A}^T\mathbf{b}$ and $\partial g(\mathbf{x}) = \theta \operatorname{sign}(\mathbf{x})$, it follows that the Forward-Backward splitting method is given by

$$\begin{aligned}\mathbf{y}^{(k+1)} &= \mathbf{x}^{(k)} - \rho\left(\mathbf{A}^T\mathbf{A}\mathbf{x}^{(k)} - \mathbf{A}^T\mathbf{b}\right) \\ \mathbf{x}^{(k+1)} &= \arg\min_{\mathbf{x}\in\mathbb{R}}\left(\|\mathbf{x}\|_1 + \frac{1}{2\rho\theta}\left\|\mathbf{x} - \mathbf{y}^{(k+1)}\right\|^2\right),\end{aligned}$$

where the final line corresponds to a soft thresholding operation, as in the case of the L1 norm minimization example in Section 2.3.2 and is thus efficient to compute.

Convergence of the algorithm follows in a similar fashion to that of both the gradient descent and proximal point methods. As this method combines both forward and backward steps we require the additional assumption that f is β-smooth to prove convergence. Like in the case of subgradient descent, this assumption ensures that ∇f is β-Lipschitz continuous and $\frac{1}{\beta}$-cocoercive.

To prove convergence, firstly, consider the squared Euclidean distance $\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\right\|^2$ where again $\mathbf{x}^* \in \mathbf{X}^*$ and $\mathbf{X}^*$ denotes the set of minimizers of (2.12). It follows that

$$\begin{aligned}\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\right\|^2 &\leq \left\|\mathbf{y}^{(k+1)} - \mathbf{y}^*\right\|^2 - \frac{1}{4}\left\|\mathbf{x}^{(k+1)} - \mathbf{y}^{(k+1)} - \left(\mathbf{x}^* - \mathbf{y}^*\right)\right\|^2 \\ &\leq \left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\|^2 + \left\|\rho\left(\partial f\left(\mathbf{x}^{(k)}\right) - \partial f\left(\mathbf{x}^*\right)\right)\right\|^2 - \left\|\mathbf{x}^{(k+1)} - \mathbf{y}^{(k+1)} - \left(\mathbf{x}^* - \mathbf{y}^*\right)\right\|^2 \\ &\quad - 2\left\langle \rho\left(\partial f\left(\mathbf{x}^{(k)}\right) - \partial f\left(\mathbf{x}^*\right)\right), \mathbf{x}^{(k)} - \mathbf{x}^*\right\rangle,\end{aligned}$$

where for the first inequality we have exploited the half averaged nature of the resolvents of maximal monotone operators. Exploiting the smoothness of f, we can use the same analysis as in Section 2.3.1 to show that that

$$\begin{aligned}\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\right\|^2 \leq \left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\|^2 &- \frac{1}{4}\left\|\mathbf{x}^{(k+1)} - \mathbf{y}^{(k+1)} - \left(\mathbf{x}^* - \mathbf{y}^*\right)\right\|^2 \\ &- \rho\left(\frac{2}{\beta} - \rho\right)\left\|\partial f\left(\mathbf{x}^{(k)}\right) - \partial f\left(\mathbf{x}^*\right)\right\|^2,\end{aligned}$$

and thus by restricting $\rho \in (0, \frac{2}{\beta})$ that $\left\|\mathbf{x}^{(k+1)} - \mathbf{y}^{(k+1)} - \left(\mathbf{x}^* - \mathbf{y}^*\right)\right\|$ and $\left\|\partial f\left(\mathbf{x}^{(k)}\right) - \partial f\left(\mathbf{x}^*\right)\right\|$ converge to 0.

In the limit where $\left\|\partial f\left(\mathbf{x}^{(k)}\right)-\partial f\left(\mathbf{x}^{*}\right)\right\|=0$, it follows that

$$\mathbf{y}^{(k+1)}-\mathbf{y}^{*}=\mathbf{x}^{(k)}-\mathbf{x}^{*}-\rho\left(f\left(\mathbf{x}^{(k)}\right)-\partial f\left(\mathbf{x}^{*}\right)\right)=\mathbf{x}^{(k)}-\mathbf{x}^{*}.$$

The residual term therefore satisfies the equality

$$\left\|\mathbf{x}^{(k+1)}-\mathbf{y}^{(k+1)}-\left(\mathbf{x}^{*}-\mathbf{y}^{*}\right)\right\|^{2}=\left\|\mathbf{x}^{(k+1)}-\mathbf{x}^{(k)}\right\|^{2},$$

and is equivalent to the fixed point residual. As $\left\|\mathbf{x}^{(k+1)}-\mathbf{x}^{(k)}\right\|^{2}$ converges to zero, this ensures that $\mathbf{x}^{(k+1)}\rightarrow\mathbf{x}\in\mathbf{X}^{*}$ and is thus optimal.

Forward-Backward splitting allows us to take advantage of the smoothness assumption of f and the firm nonexpansiveness of the resolvent operator to handle g. In particular, if the resolvent operator with respect to ∂g can be computed efficiently, this represents an attractive method of solving inclusions of sums of monotone operators. In our regularized least squares example for instance the resolvent of the L1 regularizer can be computed through its equivalence with soft thresholding which, when coupled with the analytic nature of the least squares gradient descent part, allows us to solve (2.12) in an efficient manner.

2.4.2. PEACEMAN-RACHFORD SPLITTING

While Forward-Backward splitting is a great way to take advantage of the functional characteristics of different components of our objective function, it does impose the restriction that one of our operators should be at least smooth to guarantee convergence. In practice we can easily come up with problems that don't satisfy this criterion yet were we may wish to use operator splitting to reduce the solver complexity. In this way, consider a more general prototype optimization problem given by

$$\min_{\mathbf{x}\in\mathbb{R}^{N}}\quad f(\mathbf{x})+g(\mathbf{x}), \tag{2.14}$$

where both $f,g\in\Gamma_{0}$. As with the Forward-Backward splitting example, we can form an alternate splitting technique by considering the equivalent monotonic inclusion of this problem. In particular, assuming that f and g share a common point in their domains, it follows that a minimizer of (2.14) must also satisfy the monotonic inclusion

$$\mathbf{0}\in\partial f(\mathbf{x})+\partial g(\mathbf{x}).$$

To form our alternate splitting method, we will again rephrase this problem as a fixed point inclusion as

$$\begin{aligned}
&\mathbf{0}\in\partial f(\mathbf{x})+\partial g(\mathbf{x})\iff 0\in\left(\mathbf{I}+\rho\partial f\right)(\mathbf{x})-\left(\mathbf{I}-\rho\partial g\right)(\mathbf{x})\\
&\iff 0\in\left(\mathbf{I}+\rho\partial f\right)(\mathbf{x})-\left(2\left(\mathbf{I}+\rho\partial g\right)^{-1}-\mathbf{I}\right)\circ\left(\mathbf{I}+\rho\partial g\right)(\mathbf{x})\\
&\iff 0\in\left(\mathbf{I}+\rho\partial f\right)\circ\mathbf{J}_{\partial g,\rho}(\mathbf{z})-\mathbf{R}_{\partial g,\rho}(\mathbf{z}),\ \mathbf{z}\in\left(\mathbf{I}+\rho\partial g\right)(\mathbf{x})\\
&\iff\left(\mathbf{I}+\rho\partial f\right)\circ\mathbf{J}_{\partial g,\rho}(\mathbf{z})\in\mathbf{R}_{\partial g,\rho}(\mathbf{z}),\ \mathbf{x}=\mathbf{J}_{\partial g,\rho}(\mathbf{z})\\
&\iff\left(\mathbf{I}+\rho\partial f\right)^{-1}\circ\mathbf{R}_{\partial g,\rho}\mathbf{z}=\mathbf{J}_{\partial g,\rho}\mathbf{z},\ \mathbf{x}=\mathbf{J}_{\partial g,\rho}\mathbf{z}\\
&\iff\mathbf{z}=\mathbf{R}_{\partial f,\rho}\circ\mathbf{R}_{\partial g,\rho}(\mathbf{z}),\ \mathbf{x}=\left(\mathbf{I}+\rho\partial g\right)^{-1}(\mathbf{z})
\end{aligned} \tag{2.15}$$

where in the second line we have used [47, Equation 11] and in the remainder of the derivation we have made use of the definitions of the resolvent and reflected resolvent operators. The introduced $\mathbf{z}$ variables will be referred to as *auxiliary* variables from here on out. The above splitting approach is referred to as Peaceman-Rachford splitting and recasts a monotonic inclusion of the sum of two maximal monotone operators as a fixed point inclusion based on the composition of two reflected resolvent operators.

Considering the Banach-Picard iteration of the Peaceman-Rachford fixed point inclusion, it follows that

$$\mathbf{z}^{(k+1)} = \mathbf{R}_{\partial f,\rho} \circ \mathbf{R}_{\partial g,\rho}\left(\mathbf{z}^{(k)}\right). \tag{2.16}$$

We can equivalently write this out in multiple steps by using the relationship between the resolvent and reflected resolvent as

$$\begin{aligned} \mathbf{x}^{(k+1)} &= \mathbf{J}_{\partial g,\rho}\left(\mathbf{z}^{(k)}\right), \\ \mathbf{y}^{(k+1)} &= \mathbf{J}_{\partial f,\rho}\left(2\mathbf{x}^{(k+1)} - \mathbf{z}^{(k)}\right) \\ \mathbf{z}^{(k+1)} &= 2\mathbf{y}^{(k+1)} - 2\mathbf{x}^{(k+1)} + \mathbf{z}^{(k)}. \end{aligned} \tag{2.17}$$

note that due to the single valued nature of the resolvent operator, any auxiliary fixed point vector $\mathbf{z}^*$ corresponds to an $\mathbf{x}^* \in \mathbf{X}^*$.

As we know from Section 2.2.1, the reflected resolvents of maximal monotone operators have domains which span the entire Euclidean space allowing Peaceman-Rachford splitting to be used regardless of variable initialization. Furthermore, the reflected resolvents in question are nonexpansive due to the monotonicity of ∂f and ∂g. Considering an auxiliary fixed point vector $\mathbf{z}^*$ it follows from (2.16) that the squared Euclidean distance $\left\|\mathbf{z}^{(k+1)} - \mathbf{z}^*\right\|^2$ forms a non-increasing sequence. However, as we know from Section 2.2.4, this alone is not sufficient to prove the convergence of the auxiliary variables to a fixed point.

To guarantee convergence we must impose stronger assumptions on the original functions f and g. In particular, if one function is strongly convex and differentiable, it follows that the Peaceman-Rachford splitting algorithm must converge to an auxiliary fixed point. To see this, assume that g is μ-strongly convex and differentiable. It follows that ∇g is μ-strongly monotone and single valued.

To prove convergence, we first note that the auxiliary error satisfies the inequality

$$\begin{aligned} \left\|\mathbf{z}^{(k+1)} - \mathbf{z}^*\right\|^2 &= \left\|\mathbf{R}_{\partial f,\rho} \circ \mathbf{R}_{\partial g,\rho}\left(\mathbf{z}^{(k)}\right) - \mathbf{R}_{\partial f,\rho} \circ \mathbf{R}_{\partial g,\rho}\left(\mathbf{z}^*\right)\right\|^2 \\ &\leq \left\|\mathbf{R}_{\partial g,\rho}\left(\mathbf{z}^{(k)}\right) - \mathbf{R}_{\partial g,\rho}\left(\mathbf{z}^*\right)\right\|^2 \\ &= \left\|\mathbf{z}^{(k)} - \mathbf{z}^*\right\|^2 + 4\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\right\|^2 - 4\left\langle \mathbf{z}^{(k)} - \mathbf{z}^*, \mathbf{x}^{(k+1)} - \mathbf{x}^*\right\rangle \\ &= \left\|\mathbf{z}^{(k)} - \mathbf{z}^*\right\|^2 - 4\left\langle \mathbf{z}^{(k)} - \mathbf{x}^{(k+1)} - \left(\mathbf{z}^* - \mathbf{x}^*\right), \mathbf{x}^{(k+1)} - \mathbf{x}^*\right\rangle. \end{aligned}$$

From the definition of the $\mathbf{x}$ updates in (2.17), and the differentiability of g it follows that

$$\mathbf{z}^{(k)} - \mathbf{x}^{(k+1)} = \rho \nabla g\left(\mathbf{x}^{(k+1)}\right).$$

It follows that

$$\begin{aligned}\left\|\mathbf{z}^{(k+1)}-\mathbf{z}^*\right\|^2 &\leq \left\|\mathbf{z}^{(k)}-\mathbf{z}^*\right\|^2 - 4\rho\left\langle \nabla g\left(\mathbf{x}^{(k+1)}\right)-\nabla g\left(\mathbf{x}^*\right), \mathbf{x}^{(k+1)}-\mathbf{x}^*\right\rangle \\ &\leq \left\|\mathbf{z}^{(k)}-\mathbf{z}^*\right\|^2 - 4\rho\mu\left\|\mathbf{x}^{(k+1)}-\mathbf{x}^*\right\|^2,\end{aligned}$$

where in the final line we have used the strong convexity of g. Recursively applying this inequality, the squared Euclidean error $\left\|\mathbf{x}^{(k+1)}-\mathbf{x}^*\right\|^2 \to 0$ such that the $\mathbf{x}$ variables converge to the unique optimal state $\mathbf{x}^*$ (uniqueness stems from the strong convexity of g). Similarly, due to the differentiability of g, the auxiliary iterates $\mathbf{z}^{(k)}$ must also converge to a fixed point in this instance which is unique. Unfortunately, for more general classes of functions Peaceman-Rachford splitting cannot be guaranteed to converge which again restricts the class of functions it can be used to solve.

Considering again the LASSO example given in (2.12), the Peaceman-Rachford implementation to solve this problem is given by

$$\begin{aligned}\mathbf{x}^{(k+1)} &= \left(\mathbf{I}+\rho\mathbf{A}^T\mathbf{A}\right)^{-1}\left(\mathbf{z}^{(k)}+\rho\mathbf{A}^T\mathbf{b}\right), \\ \mathbf{y}^{(k+1)} &= \arg\min_{\mathbf{x}\in\mathbb{R}}\left(\|\mathbf{x}\|_1+\frac{1}{\rho\theta}\left\|\mathbf{x}-2\mathbf{x}^{(k+1)}-\mathbf{z}^{(k)}\right\|^2\right) \\ \mathbf{z}^{(k+1)} &= 2\mathbf{y}^{(k+1)}-2\mathbf{x}^{(k+1)}+\mathbf{z}^{(k)}.\end{aligned} \tag{2.18}$$

As with the Forward-Backward splitting approach, here we again can efficiently implement the second resolvent update by noting its link with soft thresholding. The first reflected resolvent is much more expensive to implement, requiring the computation of the matrix inverse $\left(\mathbf{I}+\rho\mathbf{A}^T\mathbf{A}\right)^{-1}$ at each iteration. However, as this could be computed once and stored in memory, the additional computational cost of such an operation can be insignificant if a large number of iterations are required. Unlike Forward-Backward splitting, we no longer require g to be smooth but we do need it to be strongly convex and differentiable. For the LASSO problem considered we therefore need $\mathbf{A}$ to have full column rank such that $\frac{1}{2}\|\mathbf{Ax}-\mathbf{b}\|^2$ is strongly convex, which may not hold. In particular, sparse regularization is commonly used as a to encourage the recovery of a solution with high sparsity when solving underdetemined least squares problems. In such applications Peaceman-Rachford splitting is therefore not be a suitable choice of solver.

2.4.3. DOUGLAS-RACHFORD SPLITTING

To overcome the limitations of Peaceman-Rachford splitting, we can make a simple and minor tweak to the algorithm to guarantee convergence for all closed, convex and proper functions. In particular, by averaging the Peaceman-Rachford updates we can arrive at what is termed Douglas-Rachford splitting. We can motivate this averaging in a number of ways but perhaps the most intuitive is through an augmentation of the fixed point inclusion given in (2.15). Notably for $\mathbf{z}$ to satisfy this fixed point inclusion, the following equivalence condition must hold

$$\mathbf{z}=\mathbf{R}_{\partial f,\rho}\circ\mathbf{R}_{\partial g,\rho}(\mathbf{z}) \iff \mathbf{z}=\frac{1}{2}\left(\mathbf{I}+\mathbf{R}_{\partial f,\rho}\circ\mathbf{R}_{\partial g,\rho}\right)(\mathbf{z}).$$

In general, the same averaging procedure could be performed for any $\alpha \in (0,1)$ however, by definition, Douglas-Rachford splitting specifically refers to the half averaged case.

Applying the Banach-Picard iteration and separating out the iterates, it follows that

$$\begin{aligned}\mathbf{x}^{(k+1)} &= \mathbf{J}_{\partial g,\rho}\left(\mathbf{z}^{(k)}\right),\\ \mathbf{y}^{(k+1)} &= \mathbf{J}_{\partial f,\rho}\left(2\mathbf{x}^{(k+1)} - \mathbf{z}^{(k)}\right),\\ \mathbf{z}^{(k+1)} &= \mathbf{z}^{(k)} + \mathbf{y}^{(k+1)} - \mathbf{x}^{(k+1)}.\end{aligned}$$

The convergence of Douglas-Rachford splitting follows from the averaged nature of the operator. Namely, as we know that Peaceman-Rachford splitting is nonexpansive, it follows that

$$\left\|\mathbf{z}^{(k+1)} - \mathbf{z}^*\right\|^2 \leq \left\|\mathbf{z}^{(k)} - \mathbf{z}^*\right\|^2 - \left\|\mathbf{y}^{(k+1)} - \mathbf{x}^{(k+1)}\right\|^2.$$

Recursively applying this ensure that $\left\|\mathbf{y}^{(k+1)} - \mathbf{x}^{(k+1)}\right\|^2 \to 0$ and thus that for fixed dimension problems that $\mathbf{z}^{(k+1)} \to \mathbf{z}$ where $\mathbf{z}$ is an auxiliary fixed point.

Unlike Peaceman-Rachford splitting, the averaged nature of the Douglas-Rachford splitting requires no additional assumptions on the functions f and g other than that they are closed, convex and proper. Douglas-Rachford, and averaged versions of Peaceman-Rachford splitting in general, provide us with a method for solving monotonic inclusions of the sum of two general maximal monotone operators.

In the case of the LASSO example given in (2.12), the Douglas-Rachford implementation requires barely any modification from that given in (2.18) and is given by

$$\begin{aligned}\mathbf{x}^{(k+1)} &= \left(\mathbf{I} + \rho\mathbf{A}^T\mathbf{A}\right)^{-1}\left(\mathbf{z}^{(k)} + \rho\mathbf{A}^T\mathbf{b}\right),\\ \mathbf{y}^{(k+1)} &= \arg\min_{\mathbf{x}\in\mathbb{R}}\left(\|\mathbf{x}\|_1 + \frac{1}{\rho\theta}\left\|\mathbf{x} - 2\mathbf{x}^{(k+1)} - \mathbf{z}^{(k)}\right\|^2\right)\\ \mathbf{z}^{(k+1)} &= \mathbf{z}^{(k)} + \mathbf{y}^{(k+1)} - \mathbf{x}^{(k+1)}.\end{aligned}$$

This approach would be immediately preferred in the case that $\mathbf{A}$ is no longer full rank, where we have previously noted that Peaceman-Rachford splitting would not be guaranteed to converge.

2.5. DUALITY

In the case of the unconstrained optimization problems considered above, the basic operator splitting methods introduced thus far provide a template for efficient solver design. However, in many settings, we will typically be solving a more general form of convex optimization problems, which will most likely contain constraint functions. In particular, consider a general form optimization problem of the following form

$$\min_{\mathbf{x}\in\mathbb{R}^N} f(\mathbf{x}) \quad \mathbf{s.t.} \quad g(\mathbf{x}) \leq 0,\ \mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{0}.$$

A common approach to addressing these constraints is to exploit Lagrangian duality to rephrase this problem in the dual domain. This type of approach also has an elegant interpretation through monotone operator theory by exploiting the duality of maximal monotonicity in this instance. In the following we demonstrate a number of approaches which exploit this point to solve constrained optimization problems.

2.5.1. DUAL ASCENT

As in the case of primal domain methods, the first algorithm we will consider for solving constrained optimization problems is that of the dual ascent method. This method is essentially the dual domain equivalent of subgradient descent presented in Section 2.3.1. For this algorithm we will consider problems of the form

$$\min_{\mathbf{x}\in\mathbb{R}^N} f(\mathbf{x}) \quad \textbf{s.t.} \quad \mathbf{A}\mathbf{x}-\mathbf{b}=\mathbf{0}. \tag{2.19}$$

To solve this problem, we can equivalently consider the task of finding a saddlepoint of its associated Lagrangian [48][Chapter 5] given by

$$\mathscr{L}(\mathbf{x},\boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T(\mathbf{A}\mathbf{x}-\mathbf{b}). \tag{2.20}$$

Here, $\boldsymbol{\lambda}$ denotes the vector of dual variables associated with the constraints of (2.19). A saddlepoint of (2.20) can be found by minimizing over the primal variables. The resulting function of $\boldsymbol{\lambda}$ is referred to as the Lagrange dual function and is given by

$$g(\boldsymbol{\lambda}) = \min_{\mathbf{x}} \left(f(\mathbf{x}) + \boldsymbol{\lambda}^T\mathbf{A}\mathbf{x}\right) - \boldsymbol{\lambda}^T\mathbf{b},$$

which is a concave function of $\boldsymbol{\lambda}$. Recalling the definition of the Fenchel conjugate function f^* given in (2.1), it follows that

$$g(\boldsymbol{\lambda}) = -f^*\left(-\mathbf{A}^T\boldsymbol{\lambda}\right) - \boldsymbol{\lambda}^T\mathbf{b}.$$

To find the optimal saddlepoint, all that remains is to find the optimal $\boldsymbol{\lambda}$ by maximizing $g(\boldsymbol{\lambda})$. We can therefore solve the convex minimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^N} f^*\left(-\mathbf{A}^T\boldsymbol{\lambda}\right) + \mathbf{b}^T\boldsymbol{\lambda}, \tag{2.21}$$

which is referred to as the dual problem of (2.20).

The dual ascent algorithm solves (2.21) via gradient descent. As in the case of the subgradient descent algorithm, to prove convergence of this approach, we need to impose some functional assumptions on our original function f. Specifically, we assume that there is a feasible primal minimizer of (2.20) such that strong duality holds and furthermore that f is μ-strongly convex such that f^* is $\frac{1}{\mu}$-smooth. We can therefore use gradient descent in this dual domain to compute an optimal dual vector $\boldsymbol{\lambda}^*$. Applying gradient descent directly to this dual problem yields the iterates,

$$\boldsymbol{\lambda}^{(k+1)} = \boldsymbol{\lambda}^{(k)} + \rho\left(\mathbf{A}\nabla f^*\left(-\mathbf{A}^T\boldsymbol{\lambda}^{(k+1)}\right) - \mathbf{b}\right).$$

In general, an analytic form of f^* may not be known such that the evaluation of its gradient requires additional manipulation. Thankfully, as $\partial f^* = (\partial f)^{-1} \in \Gamma_0$ it follows that

$$\begin{aligned}\mathbf{x}^{(k+1)} = \nabla f^*\left(-\mathbf{A}^T\boldsymbol{\lambda}^{(k)}\right) &\iff 0 \in \nabla f\left(\mathbf{x}^{(k+1)}\right) + \mathbf{A}^T\boldsymbol{\lambda}^{(k)} \\ &\iff \mathbf{x}^{(k+1)} = \arg\min_{\mathbf{x}\in\mathbb{R}^N}\left(f(\mathbf{x}) + \left\langle \mathbf{A}^T\mathbf{x}-\mathbf{b}, \boldsymbol{\lambda}^{(k)}\right\rangle\right).\end{aligned}$$

The dual gradient step therefore requires first minimizing the Lagrangian of (2.19) given a current estimate of the dual variables $\boldsymbol{\lambda}$. In this way, we are able to recover an estimate of the primal variables at each iteration.

We can rewrite the dual ascent method introduced above as

$$\begin{aligned}\mathbf{x}^{(k+1)} &= \arg\min_{\mathbf{x}\in\mathbb{R}^N}\left(f(\mathbf{x}) + \left\langle \mathbf{A}^T\mathbf{x}-\mathbf{b}, \boldsymbol{\lambda}^{(k)}\right\rangle\right),\\ \boldsymbol{\lambda}^{(k+1)} &= \boldsymbol{\lambda}^{(k)} + \rho\left(\mathbf{A}\mathbf{x}^{(k+1)}-\mathbf{b}\right).\end{aligned}$$

The use of the word "ascent" in the naming of this algorithm stems from the fact that the dual update above is equivalent to a gradient ascent step where the gradient in this case stems from the Laplacian in (2.20). In actual fact however, these updates are a result of applying the classic gradient descent method to the dual problem and thus the name is somewhat misleading.

e

$$\begin{aligned}\left\|\boldsymbol{\lambda}^{(k+1)}-\boldsymbol{\lambda}^*\right\|^2 &= \left\|\boldsymbol{\lambda}^{(k)}-\boldsymbol{\lambda}^*\right\|^2 + \rho^2\left\|\mathbf{A}\nabla f^*\left(-\mathbf{A}\boldsymbol{\lambda}^{(k)}\right)-\mathbf{A}\nabla f^*\left(-\mathbf{A}\boldsymbol{\lambda}^*\right)\right\|^2\\ &\quad -2\rho\left\langle -\mathbf{A}\nabla f^*\left(-\mathbf{A}\boldsymbol{\lambda}^{(k)}\right)+\mathbf{A}\nabla f^*\left(-\mathbf{A}\boldsymbol{\lambda}^*\right), \boldsymbol{\lambda}^{(k)}-\boldsymbol{\lambda}^*\right\rangle \qquad (2.22)\\ &\leq \left\|\boldsymbol{\lambda}^{(k)}-\boldsymbol{\lambda}^*\right\|^2 - \rho\left(\frac{2\sigma^2_{\max}(\mathbf{A})}{\mu}-\rho\right)\left\|\mathbf{A}\nabla f^*\left(-\mathbf{A}\boldsymbol{\lambda}^{(k)}\right)-\mathbf{A}\nabla f^*\left(-\mathbf{A}\boldsymbol{\lambda}^*\right)\right\|^2.\end{aligned}$$

Therefore, by restricting $\rho\in(0,\frac{2\sigma^2_{\max}(\mathbf{A})}{\mu})$, the dual ascent algorithm converges to a dual optimal point. By association, as the primal variables as computed by minimizing the Lagrangian of (2.19), it follows that they also converge to a primal optimal state.

As with gradient descent, we can strengthen this result by imposing stronger functional assumptions on the function f. Notably, by making the additional assumption that f is β-smooth and that $\mathbf{A}$ has full row rank, it follows that $\mathbf{A}\nabla f^*\mathbf{A}^T$ is $\frac{\sigma^2_{\min\neq 0}(\mathbf{A})}{\beta}$-strongly monotone where $\sigma_{\min\neq 0}(\mathbf{A})$ denotes the smallest non-zeros singular value of $\mathbf{A}$. We can strengthen (2.22) so that

$$\begin{aligned}\left\|\boldsymbol{\lambda}^{(k+1)}-\boldsymbol{\lambda}^*\right\|^2 &= \left\|\boldsymbol{\lambda}^{(k)}-\boldsymbol{\lambda}^*\right\|^2 + \rho^2\left\|\mathbf{A}\nabla f^*\left(-\mathbf{A}\boldsymbol{\lambda}^{(k)}\right)-\mathbf{A}\nabla f^*\left(-\mathbf{A}\boldsymbol{\lambda}^*\right)\right\|^2\\ &\quad -2\rho\left\langle -\mathbf{A}\nabla f^*\left(-\mathbf{A}\boldsymbol{\lambda}^{(k)}\right)+\mathbf{A}\nabla f^*\left(-\mathbf{A}\boldsymbol{\lambda}^*\right), \boldsymbol{\lambda}^{(k)}-\boldsymbol{\lambda}^*\right\rangle\\ &\leq \left(1-\frac{2\rho\sigma^2_{\min\neq 0}(\mathbf{A})}{\beta}+\frac{\rho^2\sigma^4_{\max}(\mathbf{A})}{\mu^2}\right)\left\|\boldsymbol{\lambda}^{(k)}-\boldsymbol{\lambda}^*\right\|^2.\end{aligned}$$

Therefore, restricting $\rho\in\left(0,\frac{2\sigma^2_{\min\neq 0}(\mathbf{A})\mu^2}{\sigma^4_{\max}(\mathbf{A})\beta}\right)$ ensures dual convergence at a geometric rate.

2.5.2. ADMM

As in the case of primal unconstrained optimization, a combination of Lagrangian duality and monotone operator splitting can also be used to solve more complicated problems in an efficient manner. A well known and classic example of this is the alternating

direction method of multipliers (ADMM). ADMM can be used to solve convex optimization problems of the following form.

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^N} \quad & f(\mathbf{x}) + g(\mathbf{y}) \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} - \mathbf{c} = \mathbf{0}, \end{aligned} \tag{2.23}$$

where it is assumed that both functions are closed, convex and proper and that their domains contain a common feasible point. The Lagrange dual of this prototype problem is given by

$$\min_{\boldsymbol{\lambda}} \quad f^*\left(\mathbf{A}^T\boldsymbol{\lambda}\right) + g^*\left(\mathbf{B}^T\boldsymbol{\lambda}\right) - \mathbf{c}^T\boldsymbol{\lambda}. \tag{2.24}$$

Under our assumptions, both f^* and g^* are closed, convex and proper. It follows that a minimizer of (2.24) must also satisfy the monotonic inclusion

$$\mathbf{0} \in \mathbf{A}\partial f^*\left(\mathbf{A}^T\boldsymbol{\lambda}\right) + \mathbf{B}\partial g^*\left(\mathbf{B}^T\boldsymbol{\lambda}\right) - \mathbf{c}. \tag{2.25}$$

Directly applying Douglas-Rachford splitting to this problem and rearranging the resolvent operators, it follows that such a minima can be found via the algorithm

$$\begin{aligned} \mathbf{w}^{(k+1)} &= \mathbf{z}^{(k)} - \rho\left(\mathbf{A}\partial f^*\left(\mathbf{A}^T\mathbf{w}^{(k+1)}\right) - \mathbf{c}\right) \\ \boldsymbol{\nu}^{(k+1)} &= 2\mathbf{w}^{(k+1)} - \mathbf{z}^{(k)} - \rho\mathbf{B}\partial g^*\left(\mathbf{B}^T\boldsymbol{\nu}^{(k+1)}\right) \\ \mathbf{z}^{(k+1)} &= \mathbf{z}^{(k)} + \boldsymbol{\nu}^{(k+1)} - \mathbf{w}^{(k+1)}, \end{aligned} \tag{2.26}$$

where we have also exploited the single valued nature of resolvent operators. To evaluate the subdifferentials above we can utilize the same trick as in the case of dual ascent and recast these evaluations as optimization problems. We will demonstrate specifically how we can do this for the $\mathbf{w}$ updates but the same process holds for the computation of $\boldsymbol{\nu}$ as well.

Consider the computation of the iterate

$$\mathbf{w}^{(k+1)} = \mathbf{z}^{(k)} - \rho\left(\mathbf{A}f^*\left(\mathbf{A}^T\mathbf{w}^{(k+1)}\right) - \mathbf{c}\right). \tag{2.27}$$

By defining the additional variable $\mathbf{x}^{(k+1)} \in \mathbb{R}^N$ and making use of the relationship between ∂f^{-1} and ∂f^* presented in (2.1), it follows that

$$\mathbf{x}^{(k+1)} \in \partial f^*\left(\mathbf{A}^T\mathbf{w}^{(k+1)}\right) \iff \mathbf{0} \in \partial f\left(\mathbf{x}^{(k+1)}\right) - \mathbf{A}^T\mathbf{w}^{(k+1)}.$$

By then substituting this into (2.27), it follows that

$$\mathbf{w}^{(k+1)} = \mathbf{z}^{(k)} - \rho\left(\mathbf{A}\mathbf{x}^{(k+1)} - \mathbf{c}\right).$$

and thus,

$$\begin{aligned} &\mathbf{0} \in \partial f\left(\mathbf{x}^{(k+1)}\right) - \mathbf{A}^T\left(\mathbf{z}^{(k)} - \rho\left(\mathbf{A}\mathbf{x}^{(k+1)} - \mathbf{c}\right)\right) \\ &\iff \mathbf{x}^{(k+1)} = \arg\min_{\mathbf{x}\in\mathbb{R}^N}\left(f(\mathbf{x}) - \left\langle \mathbf{A}^T\mathbf{z}^{(k)}, \mathbf{x}\right\rangle + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} - \mathbf{c}\|^2\right). \end{aligned}$$

Combining this equivalent method of computation with (2.26), it follows that

$$\begin{aligned}
\mathbf{x}^{(k+1)} &= \arg\min_{\mathbf{x}\in\mathbb{R}^N}\left(f(\mathbf{x}) - \left\langle \mathbf{A}^T\mathbf{z}^{(k)}, \mathbf{x}\right\rangle + \frac{\rho}{2}\|\mathbf{A}\mathbf{x}-\mathbf{c}\|^2\right)\\
\mathbf{y}^{(k+1)} &= \arg\min_{\mathbf{y}\in\mathbb{R}^N}\left(g(\mathbf{y}) - \left\langle \mathbf{B}^T\left(\mathbf{z}^{(k)} - 2\rho\left(\mathbf{A}\mathbf{x}^{(k+1)}-\mathbf{c}\right)\right), \mathbf{y}\right\rangle + \frac{\rho}{2}\|\mathbf{B}\mathbf{y}\|^2\right)\\
\mathbf{z}^{(k+1)} &= \mathbf{z}^{(k)} - \rho\left(\mathbf{A}\mathbf{x}^{(k+1)} + \mathbf{B}\mathbf{y}^{(k+1)} - \mathbf{c}\right).
\end{aligned}$$

As noted in [47], by making the substitution $\boldsymbol{\lambda}^{(k+1)} = -\mathbf{z}^{(k+1)} + \rho\mathbf{B}\mathbf{y}^{(k+1)}$ and rearranging it follows that

$$\begin{aligned}
\mathbf{x}^{(k+1)} &= \arg\min_{\mathbf{x}\in\mathbb{R}^N}\left(f(\mathbf{x}) + \left\langle \boldsymbol{\lambda}^{(k)}, \mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{y}^{(k)}-\mathbf{c}\right\rangle + \frac{\rho}{2}\left\|\mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{y}^{(k)}-\mathbf{c}\right\|^2\right)\\
\mathbf{y}^{(k+1)} &= \arg\min_{\mathbf{y}\in\mathbb{R}^N}\left(g(\mathbf{y}) + \left\langle \boldsymbol{\lambda}^{(k)}, \mathbf{A}\mathbf{x}^{(k+1)}+\mathbf{B}\mathbf{y}-\mathbf{c}\right\rangle + \frac{\rho}{2}\left\|2\mathbf{A}\mathbf{x}^{(k+1)}+\mathbf{B}\mathbf{y}^{(k)}+\mathbf{B}\mathbf{y}-2\mathbf{c}\right\|^2\right)\\
\boldsymbol{\lambda}^{(k+1)} &= \boldsymbol{\lambda}^{(k)} + \rho\left(\mathbf{A}\mathbf{x}^{(k+1)} + \mathbf{B}\mathbf{y}^{(k)} - \mathbf{c}\right).
\end{aligned}$$

Finally, by noting that the final line has no dependence on $\mathbf{y}^{(k+1)}$, we can use the definition of $\boldsymbol{\lambda}^{(k+1)}$ and rearrange the order of computations such that

$$\begin{aligned}
\mathbf{y}^{(k+1)} &= \arg\min_{\mathbf{y}\in\mathbb{R}^N}\left(g(\mathbf{y}) + \left\langle \boldsymbol{\lambda}^{(k)}, \mathbf{A}\mathbf{x}^{(k)}+\mathbf{B}\mathbf{y}-\mathbf{c}\right\rangle + \frac{\rho}{2}\left\|\mathbf{A}\mathbf{x}^{(k)}+\mathbf{B}\mathbf{y}-\mathbf{c}\right\|^2\right)\\
\mathbf{x}^{(k+1)} &= \arg\min_{\mathbf{x}\in\mathbb{R}^N}\left(f(\mathbf{x}) + \left\langle \boldsymbol{\lambda}^{(k)}, \mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{y}^{(k+1)}-\mathbf{c}\right\rangle + \frac{\rho}{2}\left\|\mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{y}^{(k+1)}-\mathbf{c}\right\|^2\right)\\
\boldsymbol{\lambda}^{(k+1)} &= \boldsymbol{\lambda}^{(k)} + \rho\left(\mathbf{A}\mathbf{x}^{(k+1)} + \mathbf{B}\mathbf{y}^{(k+1)} - \mathbf{c}\right),
\end{aligned} \tag{2.28}$$

which demonstrates the equivalence of the Douglas-Rachford interpretation of ADMM and the augmented Lagrangian interpretation highlighted in [18].

2.5.3. PRIMAL-DUAL SPLITTING

In addition to primal and dual methods for designing convex solvers, we can combine the two approaches to develop primal-dual approaches. For instance, consider again the monotonic inclusion given in (2.25). By exploiting the relationship between ∂g^* and ∂g^{-1} we can rephrase this inclusion so that.

$$\begin{bmatrix}\mathbf{0}\\ \mathbf{x}\end{bmatrix} \in \begin{bmatrix}\mathbf{A}\partial f^*\left(\mathbf{A}^T\boldsymbol{\lambda}\right) + \mathbf{B}\mathbf{x} - \mathbf{c}\\ \partial g^*\left(\mathbf{B}^T\boldsymbol{\lambda}\right)\end{bmatrix} \iff \mathbf{0} \in \begin{bmatrix}\mathbf{A}\partial f^*\mathbf{A}^T & \mathbf{B}\\ -\mathbf{B}^T & \partial g\end{bmatrix}\begin{bmatrix}\boldsymbol{\lambda}\\ \mathbf{x}\end{bmatrix} - \begin{bmatrix}\mathbf{c}\\ \mathbf{0}\end{bmatrix}.$$

To demonstrate monotonicity we can show that the inner product

$$\begin{aligned}
&\left\langle \begin{bmatrix}\mathbf{A}\partial f^*\mathbf{A}^T & \mathbf{B}\\ -\mathbf{B}^T & \partial g\end{bmatrix}\begin{bmatrix}\boldsymbol{\lambda}_1\\ \mathbf{x}_1\end{bmatrix} - \begin{bmatrix}\mathbf{A}\partial f^*\mathbf{A}^T & \mathbf{B}\\ -\mathbf{B}^T & \partial g\end{bmatrix}\begin{bmatrix}\boldsymbol{\lambda}_2\\ \mathbf{x}_2\end{bmatrix}, \begin{bmatrix}\boldsymbol{\lambda}_1\\ \mathbf{x}_1\end{bmatrix} - \begin{bmatrix}\boldsymbol{\lambda}_2\\ \mathbf{x}_2\end{bmatrix}\right\rangle\\
&= \left\langle \mathbf{A}\partial f^*\left(\mathbf{A}^T\boldsymbol{\lambda}_1\right) - \mathbf{A}\partial f^*\left(\mathbf{A}^T\boldsymbol{\lambda}_2\right), \boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2\right\rangle + \left\langle \partial g(\mathbf{x}_1) - \partial g(\mathbf{x}_2), \mathbf{x}_1 - \mathbf{x}_2\right\rangle \geq 0,
\end{aligned}$$

where the final inequality uses the monotonicity of the subdifferentials of convex functions. It follows that this primal-dual operator is monotone. We can additionally impose

various splittings upon this operator to recover different solvers. In the case that $\mathbf{A} = \mathbf{I}$, one of the most well known primal-dual solvers is that of the Chambolle-Pock method which introduces the splitting,

$$\mathbf{0} \in \begin{bmatrix} \partial f^* & \mathbf{0} \\ -\mathbf{0} & \partial g \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} \\ \mathbf{x} \end{bmatrix} - \begin{bmatrix} \mathbf{c} \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{B} \\ -\mathbf{B}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} \\ \mathbf{x} \end{bmatrix},$$

where again, using inner products, it can be shown that both operators are monotone. The Chambolle-Pock algorithm solves this particular monotonic inclusion via the iterative sequence,

$$\boldsymbol{\lambda}^{(k+1)} = \arg\min_{\boldsymbol{\lambda}} \left(f^*(\boldsymbol{\lambda}) - \mathbf{c}^T \boldsymbol{\lambda} + \frac{1}{2} \left\| \boldsymbol{\lambda} - \left(\boldsymbol{\lambda}^{(k)} - \rho \mathbf{B}^T \mathbf{x}^{(k)} \right) \right\|^2 \right)$$
$$\mathbf{x}^{(k+1)} = \arg\min_{\mathbf{x}} \left(g(\mathbf{x}) + \frac{1}{2} \left\| \mathbf{x} - \left(\mathbf{x}^{(k)} + \rho \mathbf{B} \left(2\boldsymbol{\lambda}^{(k+1)} - \boldsymbol{\lambda}^{(k)} \right) \right) \right\|^2 \right)$$

While this method was originally proposed as distinctly different from ADMM and Douglas-Rachford splitting, with it containing a form of Douglas-Rachford splitting as a special case, it was recently demonstrated that the Chambolle-Pock method is equivalent to a special case of a lifted Douglas-Rachford splitting approach, given correct initialization. Therefore, although not demonstrated here, it is entirely possible to derive Chambolle-Pock from the perspective of monotone operator theory. For a complete version of this derivation, as well as an in-depth analysis of the consequences of this link, we refer the reader to [49].

2.6. DISTRIBUTED OPTIMIZATION

Now that we have a grasp on the application of monotone operator theory and the analysis of various convex optimization solvers, we move our attention to the main focus of this thesis; understanding and proposing methods for solving distributed convex optimization problems. In particular, we demonstrate how iterative methods derived from monotone operator theory, coupled with functional separability naturally lead to distributed solutions appropriate for use in networks.

2.6.1. CHARACTERISTICS OF DISTRIBUTED OPTIMIZATION PROBLEMS

While being the central focus of this thesis, until this point we haven't clearly defined what we mean by a *distributed solver*. In contrast to a general method for use of a single computer, we define a *distributed solver* as an algorithm which allows a network to solve a given optimization problem while only requiring local computations at each element of the network and exchanges of information between communicating computers. Specifically, a distributed solver must respect the two key attributes of *locality* and *topology* which characterize any given system.

The notion of locality is used here to refer to the inherently individual nature of elements in a network. Notably, while we desire a network to function in a cooperative manner, at a fundamental level it is comprised of unique components, each of which is equipped with its own local information and objective. In the context of an optimization problem, the local nature of such cost functions results in a naturally separable form.

Specifically, for each node i, the vector $\mathbf{x}_i \in \mathbb{R}^{M_i}$ is used to denote a local variable. Each node is also assumed to be equipped with a local cost function $f_i : \mathbb{R}^{M_i} \to \mathbb{R}$ parameterized by $\mathbf{x}_i$. The objective of a distributed optimization problem therefore contains the node separable term

$$\sum_{i \in V} f_i(\mathbf{x}_i),$$

where V denotes the set of nodes in the network. Such a cost function is appealing as it can be evaluated independently at each node without the need for communication across the network. However, this objective alone cannot facilitate collaboration as currently the nodes in the network would operate entirely independently of one another. The second notion of topology addresses this point.

The term topology refers to the physical communication structure of the underlying network. In this work we make use of graphical models to encapsulate this structure. In particular, for a given network, its graph $G(V, E)$, encodes both the set of *nodes* of the network V, as well as the set of *edges* E which captures the ability of two nodes to communicate. For any given pair of integers $i, j \in V$, the tuple $(i, j) \in E$ if and only if nodes i and j share a physical communication channel. In our context, these nodes may be computers, wireless equipped sensor nodes, autonomous cars, or whatever form the computational elements of our network may take. Figure 2.1 gives an example of such a graphical modeling process in the case of set of flight paths between cities in New Zealand. The nodes in this case are the cities themselves such as "Auckland", "Wellington" and "Christchurch" and are denoted via black circles. Key cities throughout the country can be identified via an additional gray circle around their nodes and are also labeled by name. The edges which describe the flights paths are denoted via red lines.

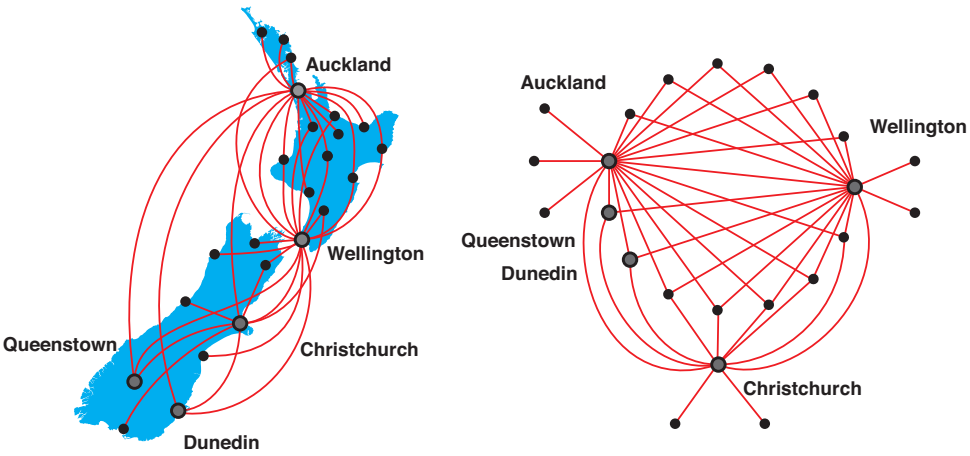


Figure 2.1: A comparison of the route map for flights within New Zealand. On the left we have the physical map of New Zealand with the flight paths overlaid while on the right we have a simple graphical model with the geographic information removed.

To facilitate collaboration within a given network, a distributed optimization problem must be comprised of more than just the local objectives at each node. However,

rather than just allowing additional terms to take on any form, in this context we require that the topology of a network restricts the manner in which the local variables at each node can interact. Specifically, if an additional cost function is to involve multiple local variables, these must share a common edge of the network. In this way, we consider a problem to be distributable if can be written as a composition of node separable and edge based terms. An example of such a prototype optimization problem is given by

$$\min_{\mathbf{x}_i \, \forall i \in V} \sum_{i \in V} f_i(\mathbf{x}_i) + \sum_{(i,j) \in E} g_{i,j}(\mathbf{x}_i, \mathbf{x}_j). \tag{2.29}$$

A special case of (2.29) commonly considered within the literature occurs when each g_{ij} is restricted to impose an affine constraint between neighboring nodes. Such constraints, which correspond to restricting the functions $g_{i|j}$ to be indicator functions can for instance be used to impose consensus agreements between neighboring nodes. The resulting optimization problem is given below

$$\min_{\mathbf{x}_i \, \forall i \in V} \sum_{i \in V} f_i(\mathbf{x}_i) \quad \text{s.t.} \, \mathbf{A}_{i|j}\mathbf{x}_i + \mathbf{A}_{j|i}\mathbf{x}_j - \mathbf{b}_{i,j} = \mathbf{0} \, \forall (i,j) \in E, \tag{2.30}$$

where the additional matrices $\mathbf{A}_{i|j}$, $\mathbf{A}_{j|i}$ and the vector $\mathbf{b}_{i,j}$ define the constraint functions referred to above. Unlike more general edge based terms, this prototype problem has an appealing structure from the perspective of solver design, as it has been reduced to a linearly constrained form. From Section 2.5, we have already seen that such problems can be readily solved using classic monotone operator splitting approaches via Lagrangian duality. Furthermore, it can be shown that the more general problem considered in (2.29) can be readily transformed to the form of (2.30) via the use of a primal lifting approach. The term primal lifting in this case refers to the act of introducing additional variables to a primal problem such as (2.29) to recast it in an equivalent form. To observe this point, consider introducing the additional variables $\mathbf{y}_{i|j}$, $\mathbf{y}_{j|i}$ for each edge of the network. By defining the neighborhood of node i by $\mathcal{N}(i) = \{j \mid (i,j) \in E\}$ and by introducing the constraints $\mathbf{x}_i = \mathbf{y}_{j|i} \; \forall \; i \in V, \; j \in \mathcal{N}(i)$, it follows that (2.29) is equivalent to

$$\min_{\mathbf{x}_i \, \forall i \in V} \sum_{i \in V} \left(f_i(\mathbf{x}_i) + \sum_{j \in \mathcal{N}(i)} \frac{1}{2} g_{i,j}(\mathbf{x}_i, \mathbf{y}_{i|j}) \right) \quad \text{s.t.} \, \mathbf{x}_i = \mathbf{y}_{j|i} \; \forall \; i \in V, \; j \in \mathcal{N}(i). \tag{2.31}$$

As the vector $\mathbf{x}_i$ and the set of vectors $\{\mathbf{y}_{i|j} \mid j \in \mathcal{N}(i)\}$ are local variables of each node i, it immediately follows that (2.31) exhibits the same form as (2.30). Therefore, by forming distributed solvers for (2.30) we can also facilitate more general distributed optimization in turn.

2.6.2. DESIGNING DISTRIBUTED SOLVERS FOR EDGE-CONSTRAINED OPTIMIZATION PROBLEMS

Despite the fact that the optimization problems considered in 2.30 exhibit a structure reflective of that of their underlying communication graph, these problems still require network-wide collaboration to achieve optimality due to the coupling of local variables at each node through the edge based constraints. The question therefore becomes how

to exploit the structure of these problems in the design of solvers that also naturally respect this underlying communication structure. The approach we will consider for this task is that of monotone operator theory.

As we have seen in this chapter thus far, monotone operator theory is an effective tool for the design and analysis of a variety of different convex solvers. Specifically, the use of operator splitting allows for the exploitation of the functional characteristics of different components of a given optimization problem to develop efficient algorithms. For instance, in Section 2.4.1 it was shown that the smooth and non-smooth components of a composite objective function could be operated on independently using operator splitting leading to a computationally efficient method of solving nonsmooth problems. Similarly, this perspective forms a natural tool for utilizing the node and edge based structure of distributed optimization problems in the construction of distributed solvers.

For the following example, reconsider the linearly constrained distributed optimization problem in (2.30) given by

$$\min_{\mathbf{x}_i \; \forall i \in V} \sum_{i \in V} f_i(\mathbf{x}_i) \quad \text{s.t.} \; \mathbf{A}_{i|j}\mathbf{x}_i + \mathbf{A}_{j|i}\mathbf{x}_j - \mathbf{b}_{i,j} = \mathbf{0} \; \forall (i,j) \in E. \tag{2.32}$$

As previously demonstrated in this chapter, monotone operator theory utilizes the subdifferentials of CCP functions, be it in a forward step or gradient descent type approach or a backward step or resolvent type operation. Importantly, the structure of our CCP functions translates through to these operations. With this in mind, consider the node separable objective of (2.32). Specifically, in addition to the ability to evaluate this function in parallel across nodes, if we consider the subdifferential of this objective with respect to the variables at any one node i, we find that

$$\frac{\partial}{\partial \mathbf{x}_i}\left(\sum_{i \in V} f_i(\mathbf{x}_i)\right) = \partial f_i(\mathbf{x}_i).$$

It follows that both any forward steps and/or backward steps involving the operator $\partial\left(\sum_{i \in V} f_i(\mathbf{x}_i)\right)$ are also parallelizable across the set of nodes. In contrast however, the constraint functions, although being all associated with physical edges of the network do not exhibit the same parallelization as each $\mathbf{x}_i$ variable is active in multiple constraints.

One way to overcome the coupling due to the constraint set, is to lift the dimension of (2.32) by introducing additional local variables $\mathbf{y}_{i|j} \; \forall \; i \in V, \; j \in \mathcal{N}(i)$ and equivalently rephrasing it as

$$\min_{\mathbf{x}_i \; \forall i \in V} \sum_{i \in V} f_i(\mathbf{x}_i) + \sum_{(i,j) \in E} \iota\left(\mathbf{y}_{i|j} + \mathbf{y}_{j|i}\right) \quad \text{s.t.} \; \mathbf{A}_{i|j}\mathbf{x}_i - \frac{\mathbf{b}_{i,j}}{2} - \mathbf{y}_{i|j} = \mathbf{0} \; \forall i \in V, \; j \in \mathcal{N}(i). \tag{2.33}$$

The introduced ι here are indicator functions defined as

$$\iota(\mathbf{x}) = \begin{cases} 0 & \text{if } \mathbf{x} = \mathbf{0} \\ +\infty & \text{otherwise} \end{cases},$$

which impose consensus constraints between the pair of variables $\mathbf{y}_{i|j}$, $\mathbf{y}_{j|i}$ associated with each edge of the network.

While being an equivalent problem, in contrast to (2.32), (2.33) now incorporates an edge separable term in its objective. This can be easily observed by noting that each pair of variables $\mathbf{y}_{i|j}$, $\mathbf{y}_{j|i}$ only play an active role in a single indicator function, which in this case are CCP functions. It follows that the subdifferential operator $\partial\left(\sum_{(i,j)\in E}\iota\left(\mathbf{y}_{i|j}-\mathbf{y}_{j|i}\right)\right)$ is able to evaluated in parallel across the set of edges. In this way, we can think of the $\mathbf{x}$ variables as being node based while the $\mathbf{y}$ variables are edge based.

At this point, we would like to note that the lifting introduced in (2.33) is but one possible approach which introduces this combination of node and edge separable terms. The main motivation for adopting this here is that it naturally leads to the formulation of a distributed solver via the ADMM algorithm derived from monotone operator theory in Section 2.5.2. Specifically, we can note that (2.33) is exactly in the form of the prototype problem in (2.23) where, in this instance,

$$f(\mathbf{x})=\sum_{i\in V}f_i(\mathbf{x}_i),\ g(\mathbf{y})=\sum_{(i,j)\in E}\iota\left(\mathbf{y}_{i|j}+\mathbf{y}_{j|i}\right).$$

By directly applying ADMM to this problem using (2.28), it follows that (2.33) can be solved via the iterative algorithm

$$\begin{aligned}
\mathbf{x}_i^{(k+1)}=&\operatorname*{arg\,min}_{\mathbf{x}_i}\left(f_i(\mathbf{x}_i)+\sum_{j\in\mathcal{N}(i)}\left\langle\boldsymbol{\lambda}_{i|j}^{(k)},\mathbf{A}_{i|j}\mathbf{x}_i-\frac{\mathbf{b}_{i,j}}{2}-\mathbf{y}_{i|j}^{(k)}\right\rangle\right.\\
&\left.+\frac{\rho}{2}\sum_{j\in\mathcal{N}(i)}\left\|\mathbf{A}_{i|j}\mathbf{x}_i-\frac{\mathbf{b}_{i,j}}{2}-\mathbf{y}_{i|j}^{(k)}\right\|^2\right)\ \forall i\in V\\
\left(\mathbf{y}_{i|j}^{(k+1)},\mathbf{y}_{j|i}^{(k+1)}\right)=&\operatorname*{arg\,min}_{\mathbf{y}_{i|j}+\mathbf{y}_{j|i}=\mathbf{0}}\left(\left\langle\boldsymbol{\lambda}_{i|j}^{(k)},\mathbf{A}_{i|j}\mathbf{x}_i^{(k+1)}-\frac{\mathbf{b}_{i,j}}{2}-\mathbf{y}_{i|j}\right\rangle+\left\langle\boldsymbol{\lambda}_{j|i}^{(k)},\mathbf{A}_{j|i}\mathbf{x}_j^{(k+1)}-\frac{\mathbf{b}_{i,j}}{2}-\mathbf{y}_{j|i}\right\rangle\right.\\
&\left.+\frac{\rho}{2}\left\|\mathbf{A}_{i|j}\mathbf{x}_i^{(k+1)}-\frac{\mathbf{b}_{i,j}}{2}-\mathbf{y}_{i|j}\right\|^2+\frac{\rho}{2}\left\|\mathbf{A}_{j|i}\mathbf{x}_j^{(k+1)}-\frac{\mathbf{b}_{i,j}}{2}-\mathbf{y}_{j|i}\right\|^2\right)\ \forall(i,j)\in E\\
\boldsymbol{\lambda}_{i|j}^{(k+1)}=&\boldsymbol{\lambda}_{i|j}^{(k)}+\rho\left(\mathbf{A}_{i|j}\mathbf{x}_i^{(k+1)}-\frac{\mathbf{b}_{i,j}}{2}-\mathbf{y}_{i|j}^{(k+1)}\right)\ \forall i\in V,\ j\in\mathcal{N}(i).
\end{aligned}\tag{2.34}$$

Furthermore, due to its linearly constrained quadratic form, the $\mathbf{y}$ updates have an analytic solution given by

$$\mathbf{y}_{i|j}^{(k+1)}=\mathbf{y}_{j|i}^{(k+1)}=\frac{1}{2}\left(\boldsymbol{\lambda}_{i|j}^{(k)}+\boldsymbol{\lambda}_{j|i}^{(k)}+\rho\left(\mathbf{A}_{i|j}\mathbf{x}_i^{(k+1)}+\mathbf{A}_{j|i}\mathbf{x}_j^{(k+1)}-\mathbf{b}_{i,j}\right)\right)$$

which corresponds to a sharing of primal $\mathbf{x}$ and dual $\boldsymbol{\lambda}$ variables across each edge of the network. The update equations of (2.34) can therefore be executed via a combination of local operations at each node and a simple data sharing action over the physical communication channels of the network.

2.6.3. DISTRIBUTED SOLVER DESIGN: BEYOND ADMM

While the distributed ADMM algorithm introduced above provides a useful means of performing distributed optimization, it is just one way of achieving this goal. Furthermore, while distributed ADMM provides a general method for solving (2.32), it is not

always the best choice of algorithm for a given application. This can be easily observed if we consider practical considerations such as the convergence rate of an algorithm. Specifically, the time taken for an algorithm to reach a desired precision has a strong bearing on its ability to be used in a practical context. For distributed approaches which exhibit some kind of dependence on the underlying topology of the network, large scale problems can therefore exhibit slow convergence. This in turn requires a large number of local computations at each node in addition to transmissions between neighboring nodes, all of which consume power. For battery powered devices, slow convergence can therefore be detrimental for the usability of an algorithm.

Recently, a method for distributed optimization termed the primal dual method of multipliers (PDMM) was proposed to solve (2.32) [50]. With regards to its practical use, it was empirically demonstrated that this method could achieve faster convergence than ADMM when applied to the same problem [51]. If such a result could be analytically shown to hold in general then clearly PDMM would be a more compelling method than ADMM. However, at the onset of the research in this thesis, PDMM was not well understood and its convergence characteristics were not quantified to argue this point one way or the other. Chapters 3 and 5 were motivated by this point and serve to both unify PDMM with monotone operator theory as well as to understand its convergence characteristics. In particular we were interested in understanding when and how PDMM may offer a competitive edge to its ADMM counterpart.

An additional limitation with distributed ADMM stems from the types of problems it can be used to solve. Specifically, while offering a means of solving (2.32), there may be target signal processing operations which cannot be easily transformed to this prototype form. The task of broadening the class of tractable distributed problems therefore represents an important avenue of research. Recent efforts within the literature reflect this point with a number of methods being proposed to solve more general prototype problems in a distributed manner [52, 53, 54, 55, 56, 57, 58, 59]. These efforts are echoed in Chapter 6 where we demonstrate a new method of distributed optimization capable of solving a broad class of separable problems, including (2.32) as a special case. This approach was derived from the perspective of monotone operator theory leading to straight forward convergence proofs based on the results introduced in this chapter.

A similar need for new algorithms arises in the context of networked systems which vary over time. In particular if the nodes within a network are allowed to move the resulting variations in topology may not be able to be handled by existing methods. For the distributed ADMM algorithm introduced in 2.34, for instance its inherent edge based nature implicitly requires the maintenance of a fixed network topology during the entire optimization process. For practical systems, this approach may therefore be infeasible to deploy. In this way, it is interesting to explore the development of new approaches which circumvent this issue. Within the literature, a number of methods have been proposed for use in time varying networks for a range of different prototype problems [10, 11, 40, 60, 61]. In Chapter 7 we contribute to this growing literature by demonstrating a new method of distributed optimization for time varying networks derived from the perspective of monotone operator theory.

2.7. A PIPELINE FOR DISTRIBUTED SIGNAL PROCESSING

The appealing solver structure demonstrated in (2.34) follows as a repercussion of the functional separability introduced in (2.33). This link provides us with a direct avenue for distributed solver design by applying a combination of problem transformation and operator splitting to form a variety of algorithms. In the case of the distributed ADMM approach introduced above for instance we exploited a combination of primal lifting, duality and Douglas-Rachford splitting to construct an ADMM solver for the considered class of linearly edge constrained distributed problems. Furthermore, this process can be combined with the standard procedure of problem transformation used to recast desired signal processing operations as convex problems to form a "pipeline" for distributed signal processing. An example of this pipeline is demonstrated in Figure 2.2.

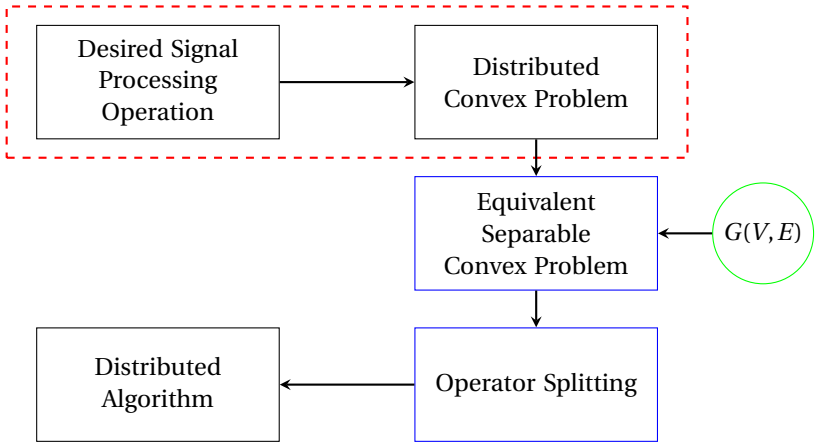


Figure 2.2: A pipeline for forming a distributed signal processing algorithm. The first two stages within the red dashed area denote the transformation of a desired signal processing problem to a convex form. The following two blue stages denote the tasks of introducing problem separability and operator splitting to form the final solver. The green circle is used in this instance to denote the dependence of the equivalent separable problem stage on the underlying topology of the target network.

The various stages of this pipeline also highlight the ways in which one can broaden the applicability of distributed signal processing to different tasks, namely via the transformation of a desired signal processing problem to a convex form, the development of new methods of separable problem transformation, and through the proposal of new operator splitting techniques. In this thesis, we pay particular attention to the first two of these stages. Specifically we focus on the broadening of basic problem classes we can solve in a distributed manner as well as the proposing of novel methods of introducing separability. At the heart of all of this, is monotone operator theory which not only provides us with a unified way to analyze different algorithms but also motivates and guides particular design choices to proposed new approaches in kind.

2.8. CONCLUSIONS

In this section we have provided a brief overview of monotone operator theory and how it can be applied to convex optimization. We have shown how a number of first order methods for both unconstrained and constrained optimization can be derived and interpreted from this perspective and have provided convergence results for these approaches. In particular, the trade off between functional assumptions and computational complexity has been highlighted and we have emphasized how operator splitting represents an effective method to exploit the summation structure of certain monotonic inclusion problems. Similarly in the case of distributed optimization, we have shown how the technique of operator splitting also provides a natural platform for the development of distributed solvers. Specifically, the combination of functional separability and the iterative nature of operator splitting methods can be exploited to generate naturally distributed algorithms which respect the topology of a given network. In particular we demonstrated how this process can be applied to form a distributed ADMM variant for linearly edge constrained problems. In the remainder of this thesis, we will explore the space of solver design and analysis following the same approach, by combining problem transformation to introduce separability and operator splitting to form a particular algorithm.

II

Analysis of Existing Distributed Solvers

3

The Primal-Dual Method of Multipliers: A Monotone Perspective

Thomas Sherson, Richard Heusdens, and W. Bastiaan Kleijn

"All have their worth and each contributes to the worth of the others."

J.R.R. Tolkien

In this chapter, we make use of the connection between monotone operator theory and convex optimization to offer a novel perspective on an existing distributed solver termed the primal dual method of multipliers (PDMM). In contrast to its initial derivation, this perspective allows us to connect PDMM with other first-order methods such as Douglas-Rachford splitting and the alternating direction method of multipliers thus providing insight into its operation. In particular, we show how PDMM combines a lifted dual form in conjunction with Peaceman-Rachford splitting to facilitate distributed optimization in undirected networks. We additionally demonstrate sufficient conditions for primal convergence for strongly convex differentiable functions and strengthen this result for strongly convex functions with Lipschitz continuous gradients by introducing a primal geometric convergence bound.

Parts of this chapter have been published in IEEE Transactions on Signal and Information Processing Over Networks, Accepted for Publication October 2018.

3.1. INTRODUCTION

The world around us is evolving through the use of large scale networking. From the way we communicate via social media [1], to the revolution of utilities and services via the paradigm of the "Internet of Things" [5], networking is reshaping the way we operate as a society. Echoing this trend, the last three decades has seen a significant rise in the deployment of large scale *sensor networks* for a wide range of applications [62, 63, 64]. Such applications include environmental monitoring [65, 66], power grid management [67, 68, 69], as well being used as part of home health care systems [70, 71].

Where centralized network topologies were once the port of call for handling data processing of sensor networks, increasingly on-node computational capabilities of such systems are being exploited to parallelize or even fully distribute data processing and computation. In contrast to their centralized counterparts such *distributed networks* have a number of distinct advantages including robustness to node failure, scalability with network size and localized transmission requirements.

Unfortunately, these distributed networks are also often characterized by limited connectivity. This limited accessibility between nodes implicitly restricts data availability making classical signal processing operations impractical or infeasible to perform. Therefore, the desire to decentralize computation requires the design of novel signal processing approaches specifically tailored to the task of in-network computation.

Within the literature, a number of methods for performing distributed signal processing have been proposed including distributed consensus [10, 12, 72], belief propagation/message passing approaches [15, 14, 13], graph signal processing over networks [73, 17, 74] and more. An additional method of particular interest to this work, is to approach the task of signal processing via its inherent connection with convex optimization. In particular, over the last two decades, it has been shown that many classical signal processing problems can be recast in an equivalent convex form [75]. By defining methods to perform distributed optimization we can therefore facilitate distributed signal processing in turn.

Recently, a new algorithm for distributed optimization called the primal dual method of multipliers (PDMM) was proposed [50]. In [50], it was shown that PDMM exhibited guaranteed average convergence, which in some examples were faster than competing methods such as the alternating direction method of multipliers (ADMM) [51]. However, there are a number of open questions surrounding the approach. In particular, prior to this work, it was unclear how PDMM was connected with similar methods within the literature.

To clarify the link between PDMM and existing works, we present a novel viewpoint of the algorithm through the lens of monotone operator theory. By demonstrating how PDMM can be derived from this perspective, we link its operation with classic operator splitting algorithms. The major strength of this observation is the fact that we can leverage results from monotone operator theory to better understand the operation of PDMM. In particular we use this insight to demonstrate new and stronger convergence results for different classes of problems than those that currently exist within the literature.

3.1.1. RELATED WORK

The work in this chapter builds upon the extensive history within the field of convex optimization in the areas of parallel and decentralized processing. In the 1970's, Rockafellar's work in network optimization [76] and the relation between convex optimization and monotone operator theory [26, 27, 28] helped establish a foundation for the field. Importantly, Rockafellar showed how linearly constrained separable convex programs can be solved in parallel via Lagrangian duality.

In the field of parallel and distributed computation, further development was undertaken by Bertsekas and Tsitsiklis [29, 30, 31] throughout the 1980's, where again separability was used as a mechanism to design a range of new algorithms. Similarly, Eckstein [32, 33] adopted an approach more reflective of Rockafellar, utilizing monotone operator theory and operator splitting to develop new distributed algorithms.

In recent years, there has been a renewed surge of interest in networked signal processing [18, 37, 38] due to the continued expansion of networked systems. This period has also seen the development of novel distributed optimization approaches for both convex and potentially non-convex problems. In the convex case, the works of [19, 20], echoing advances in three term operator splitting such as Vu-Condat splitting [77, 78], provide general frameworks for distributed convex optimization. Including classical approaches, such as ADMM, as special cases, these algorithms leverage primal-dual schemes and functional separability to create distributed implementations.

The work in [79, 80] focuses on the more general problem of potentially non-convex optimization. In particular, by at each iteration approximating both objective and constraints with specific strongly convex and smooth surrogates, the proposed methods have provable guarantees on convergence to local minima. Furthermore, in contrast to other methods, the proposed approach need not explicitly require functional separability, only the separability of the surrogates used. This allows for the optimization of problems typically outside of the scope of distributed algorithms.

3.1.2. MAIN CONTRIBUTION

The main contributions of this chapter are two-fold. Firstly we provide a novel derivation for PDMM from the perspective of monotone operator theory. In particular, we show how PDMM can be derived by combining a particular dual lifted problem with Peaceman-Rachford (PR) splitting. In contrast to its original derivation, this approach links PDMM with other classical first order methods from the literature including forward-backward splitting, Douglas-Rachford (DR) splitting and ADMM (see [81] for a recent overview).

The monotone operator perspective is also used to demonstrate a range of new convergence results for PDMM. We show how PDMM is guaranteed to converge to a primal optimal solution for strongly convex, differentiable objective functions. This result is strengthened for strongly convex functions with Lipschitz continuous gradients where a geometric convergence bound is demonstrated by linking the worst-case convergence of PDMM with that of a generalized alternating method of projections algorithm. Notably, while such results exist for PR splitting applied to dual domain optimization problems [82], they require an additional full row rank[1] assumption to ensure strong monotonic-

[1] *Row rank* refers to the dimension of the span of the row space of a matrix. *Row rank deficient* matrices have

ity which cannot be guaranteed in the case of PDMM. Furthermore, while a geometric convergence proof exists for distributed ADMM [83], currently there is no such result for PDMM. In this way the proposed work also strengthens the performance guarantees for PDMM, an important point for practical distributed optimization.

3.1.3. ORGANIZATION OF THE CHAPTER

The remainder of this chapter is organized as follows. Section 3.2 introduces appropriate nomenclature to support the manuscript. Section 3.3 introduces a monotone operator derivation of PDMM based on a specific dual lifting approach. Section 3.4 demonstrates the guaranteed primal convergence of PDMM for strongly convex and differentiable functions. This is strengthened in Section 3.5 where we demonstrate primal geometric convergence for strongly convex functions with Lipschitz continuous gradients. Finally, Section 3.6 includes simulation results to reinforce and verify the underlying claims of the document and the final conclusions are drawn in Section 3.7

3.2. NOMENCLATURE

In this work we denote by $\mathbb{R}$ the set of real numbers, by $\mathbb{R}^N$ the set of real column vectors of length N and by $\mathbb{R}^{M\times N}$ the set of M by N real matrices. Let $\mathscr{X},\mathscr{Y}\subseteq\mathbb{R}^N$. A set valued operator $\mathbf{T}:\mathscr{X}\rightarrow\mathscr{Y}$ is defined by its graph, $\text{gra}(\mathbf{T})=\{(\mathbf{x},\mathbf{y})\in\mathscr{X}\times\mathscr{Y}\mid\mathbf{y}\in\mathbf{T}(\mathbf{x})\}$. Similarly, the notion of an inverse of an operator $\mathbf{T}^{-1}$ is defined via its graph so that $\text{gra}\left(\mathbf{T}^{-1}\right)=\{(\mathbf{y},\mathbf{x})\in\mathscr{Y}\times\mathscr{X}\mid\mathbf{y}\in\mathbf{T}(\mathbf{x})\}$. $\mathbf{J}_{\mathbf{T},\rho}=\left(\mathbf{I}+\rho\mathbf{T}\right)^{-1}$ denotes the resolvent of an operator while $\mathbf{R}_{\mathbf{T},\rho}=2\mathbf{J}_{\mathbf{T},\rho}-\mathbf{I}$ denotes the reflected resolvent (Cayley operator). The fixed-point set of $\mathbf{T}$ is denoted by $\text{fix}(\mathbf{T})=\{\mathbf{x}\in\mathscr{X}\mid\mathbf{T}(\mathbf{x})=\mathbf{x}\}$. If $\mathbf{T}$ is a linear operator then $\text{ran}(\mathbf{T})$ and $\ker(\mathbf{T})$ denote its range and kernel respectively.

3.3. A DERIVATION OF THE PRIMAL-DUAL METHOD OF MULTIPLIERS BASED ON MONOTONE OPERATOR THEORY

In this section we reintroduce a recently proposed algorithm for distributed optimization termed the Primal-Dual method of multipliers (PDMM) [50]. Unlike earlier efforts within the literature [50, 51], here we demonstrate how PDMM can be derived from the perspective of monotone operator theory. In particular we show how PDMM can be derived by applying PR splitting to a certain lifted dual problem. Additionally, we highlight a previously unknown connection between PDMM and a distributed ADMM variant.

3.3.1. PROBLEM STATEMENT: NODE BASED DISTRIBUTED OPTIMIZATION

Consider an undirected network consisting of N nodes with which we want to perform convex optimization in a distributed manner. The associated graphical model of such a network is given by $\mathbf{G}(V,E)$ where $V=\{1,...,N\}$ denotes the set of nodes and E denotes the set of undirected edges so that $(i,j)\in E$ if nodes i and j share a physical connection. Note that these are simple graphs as they do not contain self loops or repeated edges. We will assume that G forms a single connected component and will denote by $\mathscr{N}(i)=$

more rows than their row rank. The notions of *column rank* and *column rank deficiency* are defined equivalently.

$\{j \in V \mid (i,j) \in E\}$ the set of neighbors of node i, i.e. those nodes j so that i and j can communicate directly. An example of such a network is given in Figure 3.1.

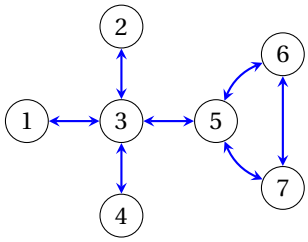


Figure 3.1: The communication graph G of a seven node network. Numbered circles denote nodes and while the arrows denote the undirected edges. The neighborhood of node five is given by the set $\mathcal{N}(5) = \{3,6,7\}$.

As previously mentioned, we are interested in using this network to perform distributed convex optimization. In this way, assume that each node i is equipped with a function $f_i \in \Gamma_0\left(\mathbb{R}^{M_i}\right)$ parameterized by a local variable $\mathbf{x}_i \in \mathbb{R}^{M_i}$. Here Γ_0 denotes the family of closed, convex and proper (CCP) functions. Under this model, consider solving the following optimization problem in a distributed manner:

$$\min_{\mathbf{x}_i \,\forall\, i \in V} \sum_{i \in V} f_i(\mathbf{x}_i) \quad \text{s.t}\, \mathbf{A}_{i|j}\mathbf{x}_i + \mathbf{A}_{j|i}\mathbf{x}_j = \mathbf{b}_{i,j} \quad \forall\, (i,j) \in E. \tag{3.1}$$

The matrices $\mathbf{A}_{i|j} \in \mathbb{R}^{M_{i,j} \times M_i}$ while the vectors $\mathbf{b}_{i,j} \in \mathbb{R}^{M_{i,j}}$. The identifier $i|j$ denotes a directed edge while i,j denotes an undirected edge. Furthermore, let $M_V = \sum_{i \in V} M_i$ and $M_E = \sum_{(i,j) \in E} M_{i,j}$. We will also assume that (3.1) is feasible. In such *distributed convex optimization problems* the terms $\mathbf{A}_{i|j}$ and $\mathbf{b}_{i,j}$ impose affine constraints between neighboring nodes.

The prototype problem in (3.1) includes, as a subset, the family of distributed consensus problems that minimize the sum of the local cost functions under network wide consensus constraints. The algorithm presented in this chapter can therefore be used for this purpose.

3.3.2. EXPLOITING SEPARABILITY VIA LAGRANGIAN DUALITY

Given the prototype problem in (3.1), the design of our distributed solver aims to address the coupling between the set of primal variables $\mathbf{x}_i$ due to the linear constraints. Echoing classic approaches in the literature, we can overcome this point via Lagrangian duality. In particular, the Lagrange dual problem of (3.1) is given by

$$\min_{\boldsymbol{\nu}} \sum_{i \in V} \left(f_i^* \left(\sum_{j \in \mathcal{N}(i)} \mathbf{A}_{i|j}^T \boldsymbol{\nu}_{i,j} \right) - \sum_{j \in \mathcal{N}(i)} \frac{\mathbf{b}_{i,j}}{2}^T \boldsymbol{\nu}_{i,j} \right), \tag{3.2}$$

where each $\boldsymbol{\nu}_{i,j} \in \mathbb{R}^{M_{i,j}}$ denotes the dual vector variable associated with the constraint at edge (i,j) and f_i^* is the Fenchel conjugate of f_i. By inspection, the resulting problem is

still separable over the set of nodes but unfortunately each $\boldsymbol{v}_{i,j}$ in (3.2) is utilized in two conjugate functions, f_i^* and f_j^*, resulting in a coupling between neighboring nodes.

To decouple the objective terms, we can lift the dimension of the dual problem by introducing copies of each $\boldsymbol{v}_{i,j}$ at nodes i and j. The pairs of additional directed edge variables are denoted by $\boldsymbol{\lambda}_{i|j}, \boldsymbol{\lambda}_{j|i}$ $\forall (i,j) \in E$ and are associated with nodes i and j respectively. To ensure equivalence of the problems, these variables are constrained so that at optimality $\boldsymbol{\lambda}_{i|j} = \boldsymbol{\lambda}_{j|i}$. The resulting problem is referred to as the *extended dual* of Eq (3.1) and is given by

$$\min_{\boldsymbol{\lambda}} \sum_{i \in V} \left(f_i^* \left(\sum_{j \in \mathcal{N}(i)} \mathbf{A}_{i|j}^T \boldsymbol{\lambda}_{i|j} \right) - \sum_{j \in \mathcal{N}(i)} \frac{\mathbf{b}_{i,j}}{2}^T \boldsymbol{\lambda}_{i|j} \right) \quad \text{s.t. } \boldsymbol{\lambda}_{i|j} = \boldsymbol{\lambda}_{j|i} \quad \forall i \in V, j \in \mathcal{N}(i).$$

The proposed lifting is appealing from the perspective of alternating minimization techniques as it partitions the resulting problem into two sections: a fully node separable objective function and a set of edge based constraints.

3.3.3. SIMPLIFICATION OF NOTATION

To assist in the derivation of our algorithm, we firstly introduce a compact vector notation for Eq. (3.3.2). Specifically we will show that (3.3.2) can be rewritten as

$$\min_{\boldsymbol{\lambda}} f^*(\mathbf{C}^T \boldsymbol{\lambda}) - \mathbf{d}^T \boldsymbol{\lambda} \quad \text{s.t } (\mathbf{I} - \mathbf{P}) \boldsymbol{\lambda} = \mathbf{0}. \tag{3.3}$$

DUAL VECTOR NOTATION

Firstly we introduce the dual variable $\boldsymbol{\lambda}$ as the stacked vector of the set of $\boldsymbol{\lambda}_{i|j}$ where the ordering of this stacking is given by $1|2 < 1|3 < \cdots < 1|N < 2|1 < 2|3 < \cdots < N|N-1$. In particular, $\boldsymbol{\lambda}$ is given by

$$\boldsymbol{\lambda} = \left[\boldsymbol{\lambda}_{1|2}^T, \cdots, \boldsymbol{\lambda}_{1|N}^T, \boldsymbol{\lambda}_{2|1}^T, \cdots, \boldsymbol{\lambda}_{N|N-1}^T \right]^T \in \mathbb{R}^{M_E}.$$

COMPACT OBJECTIVE NOTATION

Given the definition of the dual vector $\boldsymbol{\lambda}$, we now move to simplifying the objective function. Firstly, we define the sum of local functions

$$f : \mathbb{R}^{M_V} \mapsto \mathbb{R}, \mathbf{x} \mapsto \sum_{i \in V} f_i(\mathbf{x}_i),$$

where $\mathbb{R}^{M_V} = \mathbb{R}^{M_1} \times \mathbb{R}^{M_2} \times \ldots \times \mathbb{R}^{M_N}$.

We can then define a matrix $\mathbf{C} \in \mathbb{R}^{M_E \times M_V}$ and vector $\mathbf{d} \in \mathbb{R}^{M_E}$ to rewrite our objective using $\boldsymbol{\lambda}$ and f. In particular,

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_1 & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{C}_N \end{bmatrix}, \mathbf{d} = \left[\mathbf{d}_1^T, \cdots, \mathbf{d}_N^T \right]^T,$$

where the components $\mathbf{C}_i$ and $\mathbf{d}_i$ are given by

$$\mathbf{C}_i = \left[\mathbf{A}_{i|1}^T, \cdots, \mathbf{A}_{i|i-1}^T, \mathbf{A}_{i|i+1}^T, \cdots^T, \mathbf{A}_{i|N}^T\right]^T \forall i \in V,$$
$$\mathbf{d}_i = \frac{1}{2}\left[\mathbf{b}_{i,1}^T, \cdots, \mathbf{b}_{i,i-1}^T, \mathbf{b}_{i,i+1}^T, \cdots, \mathbf{b}_{i,N}^T\right]^T \forall i \in V.$$

The terms $\mathbf{A}_{i|j}$ and $\mathbf{b}_{i,j}$ are included in $\mathbf{C}_i$ and $\mathbf{d}_i$ respectively if only if $(i,j) \in E$.

The objective of Eq. (3.3.2) can therefore be rewritten as

$$f^*(\mathbf{C}^T\boldsymbol{\lambda}) - \mathbf{d}^T\boldsymbol{\lambda}.$$

COMPACT CONSTRAINTS NOTATION

Similar to the objective, we can define an additional matrix to rewrite the constraint functions using our vector notation. For this task we introduce the symmetric permutation matrix $\mathbf{P} \in \mathbb{R}^{M_E \times M_E}$ that permutes each pair of variables $\boldsymbol{\lambda}_{i|j}$ and $\boldsymbol{\lambda}_{j|i}$. This allows the constraints in (3.3.2) to be rewritten as $(\mathbf{I}-\mathbf{P})\boldsymbol{\lambda} = \mathbf{0}$. The vector $\boldsymbol{\lambda}$ is therefore only feasible if it is contained in $\ker(\mathbf{I}-\mathbf{P})$.

3.3.4. FROM THE EXTENDED DUAL PROBLEM TO A NONEXPANSIVE PDMM OPERATOR

Given the node and edge separable nature of the extended dual, we now move to forming a distributed optimization solver which takes advantage of this structure. In particular we aim to construct an operator of the form

$$\mathbf{S} = \mathbf{S}_E \circ \mathbf{S}_N,$$

where $\mathbf{S}_N$ and $\mathbf{S}_E$ are parallelizable over the nodes and edges respectively and $\circ$ is used to denote their composition so that $\forall\,(\mathbf{x},\mathbf{z}) \in \mathrm{gra}\,(\mathbf{S}_1 \circ \mathbf{S}_2)\,, \exists \mathbf{y} |\,(\mathbf{x},\mathbf{y}) \in \mathrm{gra}\,(\mathbf{S}_1)\,, (\mathbf{y},\mathbf{z}) \in \mathrm{gra}\,(\mathbf{S}_2)$. Furthermore, we would like such operators to be nonexpansive so that classic iterative solvers can be employed. The nonexpansiveness of an operator is defined as follows.

Definition 3.3.1. *Nonexpansive Operators: An operator* $\mathbf{T}: \mathscr{X} \to \mathscr{Y}$ *is nonexpansive if*

$$\forall\,(\mathbf{x},\mathbf{u})\,,(\mathbf{y},\mathbf{v}) \in \mathrm{gra}\,(\mathbf{T}) : \|\mathbf{u}-\mathbf{v}\| \leq \|\mathbf{x}-\mathbf{y}\|,$$

We can construct such an $\mathbf{S}$ by making use of the relationship between monotone operators and the subdifferentials of convex functions. In particular, an operator is monotone if it satisfies the following definition.

Definition 3.3.2. *Monotone Operators: An operator* $\mathbf{T}: \mathscr{X} \to \mathscr{Y}$ *is monotone if*

$$\forall\,(\mathbf{x},\mathbf{u})\,,(\mathbf{y},\mathbf{v}) \in \mathrm{gra}\,(\mathbf{T}) : \langle \mathbf{u}-\mathbf{v}, \mathbf{x}-\mathbf{y}\rangle \geq 0,$$

Furthermore, $\mathbf{T}$ *is maximal monotone if*

$$\textit{there does not exist a monotone } \tilde{\mathbf{T}}: \mathscr{X} \to \mathscr{Y} \mid \mathrm{gra}(\mathbf{T}) \subset \mathrm{gra}(\tilde{\mathbf{T}}).$$

With these definitions in mind, consider the equivalent unconstrained form of (3.3) given by

$$\min_{\boldsymbol{\lambda}} \quad f^*(\mathbf{C}^T\boldsymbol{\lambda}) - \mathbf{d}^T\boldsymbol{\lambda} + \iota_{\ker(\mathbf{I}-\mathbf{P})}(\boldsymbol{\lambda}), \tag{3.4}$$

where $\iota_{\ker(\mathbf{I}-\mathbf{P})}$ is an indicator function defined as

$$\iota_{\ker(\mathbf{I}-\mathbf{P})}(\mathbf{y}) = \begin{cases} 0 & (\mathbf{I}-\mathbf{P})\mathbf{y} = \mathbf{0} \\ +\infty & \text{otherwise.} \end{cases}$$

As $\ker(\mathbf{I}-\mathbf{P})$ is a closed subspace, it follows from [34, Example 1.25] that $\iota_{\ker(\mathbf{I}-\mathbf{P})} \in \Gamma_0$. Furthermore, as $f \in \Gamma_0$, using [34, Theorem 13.32, Prop. 13.11], it follows that $f^*\left(\mathbf{C}^T\right) \in \Gamma_0$ as well. Due to our feasibility assumption of (3.1), the relative interiors of the domains of $f^*\left(\mathbf{C}^T\right)$ and $\iota_{\ker(\mathbf{I}-\mathbf{P})}$ share a common point. From [34, Theorem 16.3], it follows that $\boldsymbol{\lambda}^*$ is a minimizer of (3.4) if and only if

$$\mathbf{0} \in \mathbf{C}\partial f^*\left(\mathbf{C}^T\boldsymbol{\lambda}^*\right) - \mathbf{d} + \partial\iota_{\ker(\mathbf{I}-\mathbf{P})}\left(\boldsymbol{\lambda}^*\right). \tag{3.5}$$

Note that the operators $\mathbf{T}_1 = \mathbf{C}\partial f^*\mathbf{C}^T - \mathbf{d}$ and $\mathbf{T}_2 = \partial\iota_{\ker(\mathbf{I}-\mathbf{P})}$ are by design separable over the set of nodes and edges respectively. Furthermore, $\mathbf{C}\partial f^*\mathbf{C}^T$ and $\partial\iota_{\ker(\mathbf{I}-\mathbf{P})}$ are the subdifferentials of CCP functions and thus are maximal monotone. A zero-point of (3.5) can therefore be found via a range of operator splitting methods (see [32] for an overview).

In this particular instance, we will use PR splitting to construct a nonexpansive PDMM operator by rephrasing the zero-point condition in (3.5) as a more familiar fixed-point condition. This equivalent condition, as demonstrated in [47] (Section 7.3), is given by

$$\mathbf{R}_{\mathbf{T}_2,\rho} \circ \mathbf{R}_{\mathbf{T}_1,\rho}(\mathbf{z}) = \mathbf{z}, \ \boldsymbol{\lambda} = \mathbf{J}_{\mathbf{T}_1,\rho}(\mathbf{z}),$$

where $\mathbf{R}_{\mathbf{T}_i,\rho}$ and $\mathbf{J}_{\mathbf{T}_i,\rho}$ are the reflected resolvent and resolvent operators of $\mathbf{T}_i$ respectively. Here, the introduced $\mathbf{z}$ variables will be referred to as an *auxiliary* variables.

We define the PDMM operator as

$$\mathbf{T}_{P,\rho} = \mathbf{R}_{\mathbf{T}_2,\rho} \circ \mathbf{R}_{\mathbf{T}_1,\rho},$$

which will be used repeatedly throughout this work. Importantly given the nature of the operators considered, $\mathbf{T}_{P,\rho}$ is nonexpansive. Specifically, as both $\mathbf{T}_1$ and $\mathbf{T}_2$ are maximal monotone operators, $\mathbf{J}_{\mathbf{T}_1,\rho}$ and $\mathbf{J}_{\mathbf{T}_2,\rho}$ are both firmly nonexpansive. By [34, Proposition 4.2], it follows that $\mathbf{R}_{\mathbf{T}_1,\rho}$ and $\mathbf{R}_{\mathbf{T}_2,\rho}$ are nonexpansive. The nonexpansiveness of $\mathbf{T}_{P,\rho}$ allows us to utilize fixed-point iterative methods to solve (3.3.2) and ultimately (3.1) in a distributed manner.

3.3.5. ON THE LINK WITH THE PRIMAL DUAL METHOD OF MULTIPLIERS

We now demonstrate how PDMM, as defined in [50], can be linked with classical monotone operator splitting theory. For this purpose we will consider the fixed-point iteration of $\mathbf{T}_{P,\rho}$ given by

$$\mathbf{z}^{(k+1)} = \mathbf{T}_{P,\rho}\left(\mathbf{z}^{(k)}\right) = \mathbf{R}_{\mathbf{T}_2,\rho} \circ \mathbf{R}_{\mathbf{T}_1,\rho}\left(\mathbf{z}^{(k)}\right). \tag{3.6}$$

To aid in the aforementioned relationship, the evaluation of the reflected resolvent operators $\mathbf{R}_{\mathbf{T}_1,\rho}$ and $\mathbf{R}_{\mathbf{T}_2,\rho}$ are outlined in the following Lemmas.

Lemma 3.3.1. $\mathbf{y}^{(k+1)} = \mathbf{R}_{\mathbf{T}_1,\rho}\left(\mathbf{z}^{(k)}\right)$ *can be computed as*

$$\begin{aligned}\mathbf{x}^{(k+1)} =& \arg\min_{\mathbf{x}}\left(f(\mathbf{x}) - \langle \mathbf{C}^T\mathbf{z}^{(k)}, \mathbf{x}\rangle + \frac{\rho}{2}||\mathbf{C}\mathbf{x}-\mathbf{d}||^2\right)\\ \boldsymbol{\lambda}^{(k+1)} =& \mathbf{z}^{(k)} - \rho\left(\mathbf{C}\mathbf{x}^{(k+1)} - \mathbf{d}\right)\\ \mathbf{y}^{(k+1)} =& 2\boldsymbol{\lambda}^{(k+1)} - \mathbf{z}^{(k)}.\end{aligned}$$

A proof of this result can be found in Appendix 3.A. Note that the block diagonal structure of $\mathbf{C}$ and the separability of f allow this reflected resolvent to be computed in parallel across the nodes.

Lemma 3.3.2. $\mathbf{z}^{(k+1)} = \mathbf{R}_{\mathbf{T}_2,\rho}\left(\mathbf{y}^{(k+1)}\right)$ *can be computed as* $\mathbf{z}^{(k+1)} = \mathbf{P}\mathbf{y}^{(k+1)}$.

The proof for this result is included in Appendix 3.B. The resulting permutation operation is equivalent to an exchange of auxiliary variables between neighboring nodes and is therefore distributable over the underlying network.

Utilizing Lemmas 3.3.1 and 3.3.2 it follows that

$$\mathbf{T}_{P,\rho} = \mathbf{P}\circ\mathbf{R}_{\mathbf{T}_1,\rho}, \tag{3.7}$$

and thus that (3.6) is equivalent to

$$\mathbf{z}^{(k+1)} = \mathbf{P}\left(\mathbf{z}^{(k)} - 2\rho\left(\mathbf{C}\mathbf{x}^{(k+1)} - \mathbf{d}\right)\right). \tag{3.8}$$

By noting that $\mathbf{z}^{(k+1)} = \mathbf{P}\left(\boldsymbol{\lambda}^{(k+1)} - \rho\left(\mathbf{C}\mathbf{x}^{(k+1)} - \mathbf{d}\right)\right)$, the dependence on $\mathbf{y}^{(k+1)}$ and $\mathbf{z}^{(k+1)}$ can be removed, reducing the scheme to that given in Algorithm 1.

Algorithm 1 Simplified PDMM

1: **Initialise:** $\boldsymbol{\lambda}^{(0)} \in \mathbb{R}^{M_E}$, $\mathbf{x}^{(0)} \in \mathbb{R}^{M_V}$
2: **for** k=0,..., **do**
3: $\quad\mathbf{x}^{(k+1)} = \arg\min_{\mathbf{x}}\left(f(\mathbf{x}) - \langle \mathbf{C}^T\mathbf{P}\boldsymbol{\lambda}^{(k)}, \mathbf{x}\rangle + \frac{\rho}{2}||\mathbf{C}\mathbf{x} + \mathbf{P}\mathbf{C}\mathbf{x}^{(k)} - 2\mathbf{d}||^2\right)$
4: $\quad\boldsymbol{\lambda}^{(k+1)} = \mathbf{P}\boldsymbol{\lambda}^{(k)} - \rho\left(\mathbf{C}\mathbf{x}^{(k+1)} + \mathbf{P}\mathbf{C}\mathbf{x}^{(k)} - 2\mathbf{d}\right)$
5: **end for**

This algorithm is identical to a particular instance of PDMM proposed in [50]. Thus, PDMM is equivalent to the fixed-point iteration of the PR splitting of the extended dual problem, linking the approach with a plethora of existing algorithms within the literature [18, 20, 84, 85].

The connection with PR splitting motivates why PDMM may converge faster than ADMM for some problems, as demonstrated in [50]. In particular, [82, Remark 4] notes that PR splitting provides the fastest bound on convergence even though it may not converge for general problems. Specifically, the strong convexity and Lipschitz continuity of the gradient of the averaging problem considered in [50] supports this link.

Algorithm 2 Distributed PDMM

Initialise: $\mathbf{z}^{(0)} \in \mathbb{R}^{M_E}$
for k=0,..., **do**
  for all $i \in V$ **do** ▷ Primal Update
    $\mathbf{x}_i^{(k+1)} = \arg\min_{\mathbf{x}_i} \left(f_i(\mathbf{x}_i) + \sum_{j \in \mathcal{N}(i)} \left(-\langle \mathbf{A}_{i|j}^T \mathbf{z}_{i|j}^{(k)}, \mathbf{x}_i \rangle + \frac{\rho}{2} ||\mathbf{A}_{i|j}\mathbf{x}_i - \frac{\mathbf{b}_{i,j}}{2}||^2 \right) \right)$
    for all $j \in \mathcal{N}(i)$ **do** ▷ Dual Update
      $\mathbf{y}_{i|j}^{(k+1)} = \mathbf{z}_{i|j}^{(k)} - 2\rho \left(\mathbf{A}_{i|j}\mathbf{x}_i^{(k+1)} - \frac{\mathbf{b}_{i,j}}{2} \right)$
    end for
  end for
  for all $i \in V, j \in \mathcal{N}(i)$ **do** ▷ Transmit Variables
    $\mathbf{Node}_j \leftarrow \mathbf{Node}_i(\mathbf{y}_{i|j}^{(k+1)})$
  end for
  for all $i \in V, j \in \mathcal{N}(i)$ **do** ▷ Auxiliary Update
    $\mathbf{z}_{i|j}^{(k+1)} = \mathbf{y}_{j|i}^{(k+1)}$
  end for
end for

The distributed nature of PDMM can be more easily visualized in Algorithm 2 where we have utilized the definitions of $\mathbf{C}$ and $\mathbf{d}$. Here the notation $\mathbf{Node}_j \leftarrow \mathbf{Node}_i(\bullet)$ indicates the transmission of data from node i to node j.

Each iteration of the algorithm only requires one-way transmission of the auxiliary $\mathbf{z}$ variables between neighboring nodes. Thus, no direct collaboration is required between nodes during the computation of each iteration leading to an appealing mode of operation for use in practical networks.

3.3.6. ON THE LINK WITH THE DISTRIBUTED ALTERNATING DIRECTION METHOD OF MULTIPLIERS

Using the proposed monotone interpretation of PDMM we can also link its behavior with ADMM. While in [50] it was suggested that these two methods were fundamentally different due to their contrasting derivations, in the following we demonstrate how they are more closely related than first thought. Interestingly, this link is masked via the change of variables typically used in the updating scheme for ADMM and PDMM (see [18, Section 3] and [50, Section 4] respectively for such representations). For this purpose we re-derive an ADMM variant from the perspective of monotone operator theory.

To begin, consider the prototype ADMM problem given by

$$\min_{\mathbf{x},\mathbf{y}} f(\mathbf{x}) + g(\mathbf{y}) \quad \text{s.t. } \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} = \mathbf{c}. \tag{3.9}$$

We can recast (3.1), in the form of (3.9) by introducing the additional variables $\mathbf{y}_{i|j}, \mathbf{y}_{j|i} \in$

$\mathbb{R}^{M_{i,j}}$ $\forall (i,j) \in E$ so that

$$\min_{\mathbf{x}} \sum_{i \in V} f_i(\mathbf{x}_i) \quad \text{s.t} \begin{array}{l} \mathbf{A}_{i|j}\mathbf{x}_i - \frac{\mathbf{b}_{i,j}}{2} = \mathbf{y}_{i|j} \\ \mathbf{A}_{j|i}\mathbf{x}_j - \frac{\mathbf{b}_{i,j}}{2} = \mathbf{y}_{j|i} \\ \mathbf{y}_{i|j} + \mathbf{y}_{j|i} = \mathbf{0} \end{array} \Bigg\} \forall (i,j) \in E. \tag{3.10}$$

Defining the stacked vector $\mathbf{y} \in \mathbb{R}^{M_E}$ and adopting the matrices $\mathbf{C}$, $\mathbf{P}$ and $\mathbf{d}$ as per Section 3.3.3, (3.10) can be more simply written as

$$\min_{\mathbf{x}} f(\mathbf{x}) + \iota_{\ker(\mathbf{I}+\mathbf{P})}(\mathbf{y}) \quad \text{s.t}\, \mathbf{C}\mathbf{x} - \mathbf{d} = \mathbf{y}. \tag{3.11}$$

Here, the indicator function is used to capture the final set of equality constraints in (3.10). It follows that (3.11) is exactly in the form of (3.9) so that ADMM can be applied.

The ADMM algorithm is equivalent to applying Douglas Rachford (DR) splitting [86] to the dual of (3.11), given by

$$\min_{\boldsymbol{\lambda}_i \, \forall \, i \in V} \quad f^*\left(\mathbf{C}^T\boldsymbol{\lambda}\right) - \mathbf{d}^T\boldsymbol{\lambda} + \iota^*_{\ker(\mathbf{I}+\mathbf{P})}(\boldsymbol{\lambda}), \tag{3.12}$$

where $\boldsymbol{\lambda}$, as in the case of PDMM, denotes the stacked vector of dual variables associated with the directed edges.

Comparing (3.12) and (3.5), we can note that the apparent difference in the dual problems is due to the use of $\iota_{\ker(\mathbf{I}-\mathbf{P})}$, in the case of PDMM, or $\iota^*_{\ker(\mathbf{I}+\mathbf{P})}$ in the case of ADMM. In actual fact these two functions are equal which stems from the definition of the Fenchel conjugate of an indicator function,

$$\iota^*_{\ker(\mathbf{I}+\mathbf{P})}(\boldsymbol{\lambda}) = \sup_{\mathbf{y}} \left(\langle \mathbf{y}, \boldsymbol{\lambda} \rangle - \iota_{\ker(\mathbf{I}+\mathbf{P})}(\mathbf{y}) \right) = \begin{cases} 0 & \boldsymbol{\lambda} \in \operatorname{ran}(\mathbf{I}+\mathbf{P}) \\ \infty & \text{otherwise.} \end{cases}$$

As $\operatorname{ran}(\mathbf{I}+\mathbf{P}) = \ker(\mathbf{I}-\mathbf{P})$, it follows that $\iota^*_{\ker(\mathbf{I}+\mathbf{P})} = \iota_{\ker(\mathbf{I}-\mathbf{P})}$. The problems in (3.4) and (3.12) are therefore identical.

As DR splitting is equivalent to a half averaged form of PR splitting [34], the operator form of ADMM is therefore given by $\mathbf{T}_{A,\rho} = \frac{1}{2}\left(\mathbf{I} + \mathbf{T}_{P,\rho}\right)$. In this manner, despite their differences in earlier derivations, ADMM and PDMM are fundamentally linked. Within the literature, PDMM could therefore also be referred to as a particular instance of *generalised* [87] or *relaxed* ADMM [82].

3.4. GENERAL CONVERGENCE RESULTS FOR PDMM

Having linked PDMM with PR splitting, we now move to demonstrate convergence results for the algorithm. In particular we demonstrate a proof of convergence for PDMM for strongly convex and differentiable functions. This proof is required due to the fact that the strong monotonicity of either $\mathbf{T}_1$ or $\mathbf{T}_2$, usually required to guarantee convergence of PR splitting, cannot be guaranteed for PDMM due to the row rank deficiency of the matrix $\mathbf{C}$. We also highlight the use of operator averaging to guarantee convergence for all $f \in \Gamma_0$ and demonstrate its necessity with an analytic example where PDMM fails to converge.

3.4.1. CONVERGENCE OF THE PRIMAL ERROR ($\|\mathbf{x}^{(k)}-\mathbf{x}^*\|^2$) OF PDMM

The first result we demonstrate is that of the *primal convergence* of PDMM. In particular, we show that the sequence of primal iterates $\left(\mathbf{x}^{(k)}\right)_{k\in\mathbb{N}}$ converges to an optimal state, i.e.,

$$\exists \mathbf{x}^* \in \mathbf{X}^* \mid \|\mathbf{x}^{(k)}-\mathbf{x}^*\|^2 \to 0. \tag{3.13}$$

where $\mathbf{X}^*$ denotes the set of primal optimizers of (3.1) and $\bullet \to \bullet$ denotes convergence. The term $\|\mathbf{x}^{(k)}-\mathbf{x}^*\|^2$ will be referred to as the *primal error* from here on.

Many of the arguments used in this section make use of the properties of the kernel and range matrices. These are defined below.

Definition 3.4.1. *Range Space and Kernel Space: Given a matrix* $\mathbf{A}$*, the range space of* $\mathbf{A}$ *is denoted by* $\text{ran}(\mathbf{A})$ *where*

$$\forall \mathbf{y} \in \text{ran}(\mathbf{A}), \exists \mathbf{u} \mid \mathbf{A}\mathbf{u} = \mathbf{y}.$$

Similarly, the kernel space of $\mathbf{A}$ *is denoted by* $\ker(\mathbf{A})$ *where*

$$\forall \mathbf{y} \in \ker(\mathbf{A}), \mathbf{A}\mathbf{y} = \mathbf{0}.$$

For any matrix, the subspaces $\text{ran}(\mathbf{A})$ *and* $\ker\left(\mathbf{A}^T\right)$ *are orthogonal and, furthermore, their direct sum* $\text{ran}(\mathbf{A}) + \ker\left(\mathbf{A}^T\right)$ *spans the entire space.*

To demonstrate that (3.13) holds, we can make use of the relationship between the primal $\mathbf{x}$ and auxiliary $\mathbf{z}$ variables of PDMM. In particular, we will demonstrate that both the primal and auxiliary variables converge by ultimately showing that

$$\exists \mathbf{z}^* \in \text{fix}\left(\mathbf{T}_{P,\rho}\right) \mid \|\mathbf{z}^{(k)}-\mathbf{z}^*\|^2 \to 0,$$

which we will refer to as *auxiliary convergence*.

3.4.2. PRIMAL INDEPENDENCE OF A NON-DECREASING SUBSPACE

To prove auxiliary convergence, other approaches in the literature often leverage additional operational properties such as strict nonexpansiveness. Unfortunately, in the case of PDMM, $\mathbf{T}_{P,\rho}$ is at best nonexpansive due to the presence of a non-decreasing component. Fortunately, this particular component does not influence the computation of the primary variables and ultimately can be ignored.

To demonstrate that PDMM is at best nonexpansive, consider the equation for two successive updates given by

$$\begin{aligned}\mathbf{z}^{(k+2)} &= \mathbf{T}_{P,\rho} \circ \mathbf{T}_{P,\rho}\left(\mathbf{z}^{(k)}\right) \\ &= \mathbf{T}_{P,\rho}\left(\mathbf{P}\left(\mathbf{z}^{(k)} - 2\rho\left(\mathbf{C}\mathbf{x}^{(k+1)} - \mathbf{d}\right)\right)\right) \\ &= \mathbf{z}^{(k)} - 2\rho\left(\mathbf{P}\mathbf{C}\mathbf{x}^{(k+2)} + \mathbf{C}\mathbf{x}^{(k+1)} - 2\mathbf{d}\right),\end{aligned} \tag{3.14}$$

where the second and third lines use the PDMM update in (3.8). From our feasibility assumption of (3.1), $\exists \mathbf{x}^* \mid \mathbf{P}\mathbf{C}\mathbf{x}^* + \mathbf{C}\mathbf{x}^* = 2\mathbf{d}$ so that $\mathbf{d} \in \text{ran}(\mathbf{P}\mathbf{C}) + \text{ran}(\mathbf{C})$. Therefore, every two PDMM updates only affect the auxiliary variables in the subspace $\text{ran}(\mathbf{P}\mathbf{C}) + \text{ran}(\mathbf{C})$.

By considering the projection of each iterate onto the orthogonal subspace of ran $(\mathbf{PC}) +$ ran $(\mathbf{C})$, which is given by $\ker\left(\mathbf{C}^T\right) \cap \ker\left(\mathbf{C}^T\mathbf{P}\right)$, it follows that, for all even k,

$$\begin{aligned}\prod_{\ker(\mathbf{C}^T)\cap\ker(\mathbf{C}^T\mathbf{P})}\left(\mathbf{z}^{(k+2)}\right) &= \prod_{\ker(\mathbf{C}^T)\cap\ker(\mathbf{C}^T\mathbf{P})}\left(\mathbf{z}^{(k)}\right)\\ &= \prod_{\ker(\mathbf{C}^T)\cap\ker(\mathbf{C}^T\mathbf{P})}\left(\mathbf{z}^{(0)}\right),\end{aligned}$$

where $\prod_{\mathscr{A}}$ denotes the orthogonal projection onto $\mathscr{A}$.

Every even-numbered auxiliary iterate $\mathbf{z}^{(k)}$ contains a non-decreasing component determined by our initial choice of $\mathbf{z}^{(0)}$. Fortunately, from Lemma 3.3.1 it is clear that each $\mathbf{x}^{(k)}$ is independent of $\prod_{\ker(\mathbf{C}^T)}\left(\mathbf{z}^{(k)} + \rho\mathbf{d}\right)$. As $\ker\left(\mathbf{C}^T\right) \cap \ker\left(\mathbf{C}^T\mathbf{P}\right) \subseteq \ker\left(\mathbf{C}^T\right)$, any signal in the non-decreasing subspace of $\mathbf{T}_{P,\rho} \circ \mathbf{T}_{P,\rho}$ will not play a role in the primal updates. For proving primal convergence, we will therefore consider the *projected auxiliary error*

$$\| \prod_{\text{ran}(\mathbf{C})+\text{ran}(\mathbf{PC})}\left(\mathbf{z}^{(k)} - \mathbf{z}^*\right) \|^2. \tag{3.15}$$

Such a projection can be easily computed for even iterates due to the structure noted in (3.14) by defining the vector

$$\mathbf{z}^\diamond = \mathbf{z}^* + \prod_{\ker(\mathbf{C}^T)\cap\ker(\mathbf{C}^T\mathbf{P})}\left(\mathbf{z}^{(0)}\right). \tag{3.16}$$

From the nonexpansiveness of PDMM, the projected auxiliary error satisfies

$$\|\mathbf{z}^{(k+2)} - \mathbf{z}^\diamond\| \leq \|\mathbf{z}^{(k)} - \mathbf{z}^\diamond\|.$$

The sequence $\left(\mathbf{z}^{(2k)}\right)_{k\in\mathbb{N}}$ is therefore Fejér monotone with respect to $\mathbf{z}^\diamond$ and thus the sequence $\left(\|\mathbf{z}^{(2k)} - \mathbf{z}^\diamond\|\right)_{k\in\mathbb{N}}$ converges [34, Proposition 5.4]. To prove projected auxiliary convergence, all that remains is to show that

$$\lim_{k\to\infty}\left(\mathbf{z}^{(2k)} - \mathbf{z}^\diamond\right) = 0. \tag{3.17}$$

3.4.3. OPTIMALITY OF AUXILIARY LIMIT POINTS

We will now demonstrate that (3.17) holds in the specific case of strongly convex and differentiable functions, in turn allowing us to prove primal convergence. While the differentiability of a function is straightforward, the notion of strong convexity is defined below.

Definition 3.4.2. *Strong Convexity: A function f is μ-strongly convex with $\mu > 0$ if for all $\theta \in [0,1], \mathbf{x}, \mathbf{y} \in dom(f)$,*

$$f\left(\theta\mathbf{x} + (1-\theta)\mathbf{y}\right) \leq \theta f(\mathbf{x}) + (1-\theta) f\left(\mathbf{y}\right) - \mu\theta(1-\theta)\|\mathbf{x}-\mathbf{y}\|^2.$$

Additionally, if f is μ-strongly convex, ∂f is μ-strongly monotone.

Definition 3.4.3. *Strongly Monotone: An operator* $\mathbf{T}:\mathscr{X}\rightarrow\mathscr{Y}$ *is* μ*-strongly monotone with* $\mu>0$*, if*

$$\langle \mathbf{u}-\mathbf{v},\mathbf{x}-\mathbf{y}\rangle \geq \mu\|\mathbf{x}-\mathbf{y}\|^2 \quad \forall\,(\mathbf{x},\mathbf{u}),(\mathbf{y},\mathbf{v})\in \operatorname{gra}(\mathbf{T}).$$

To verify that (3.17) holds under the aforementioned assumptions, we make use of the following Lemma relating to the limit points of the primal and dual variables.

Lemma 3.4.1. *If* f *is differentiable and* μ*-strongly convex then*

$$\begin{aligned}\lim_{k\rightarrow\infty}\mathbf{x}^{(k)}&=\mathbf{x}^*,\\ \lim_{k\rightarrow\infty}\prod_{\operatorname{ran}(\mathbf{C})}\left(\boldsymbol{\lambda}^{(k)}\right)&=\prod_{\operatorname{ran}(\mathbf{C})}\left(\boldsymbol{\lambda}^*\right).\end{aligned}$$

The proof for this Lemma can be found in Appendix 3.C.

Using Lemma 3.4.1, and rearranging the dual update equation in Lemma 3.3.1, it follows that

$$\begin{aligned}\lim_{k\rightarrow\infty}\prod_{\operatorname{ran}(\mathbf{C})}\left(\mathbf{z}^{(k)}\right)&=\lim_{k\rightarrow\infty}\prod_{\operatorname{ran}(\mathbf{C})}\left(\boldsymbol{\lambda}^{(k+1)}+\rho\left(\mathbf{C}\mathbf{x}^{(k+1)}-\mathbf{d}\right)\right)\\ &=\prod_{\operatorname{ran}(\mathbf{C})}\left(\boldsymbol{\lambda}^*+\rho\left(\mathbf{C}\mathbf{x}^*-\mathbf{d}\right)\right)=\prod_{\operatorname{ran}(\mathbf{C})}\left(\mathbf{z}^*\right).\end{aligned}\tag{3.18}$$

From (3.18), if also follows that

$$\begin{aligned}\mathbf{0}&=\lim_{k\rightarrow\infty}\prod_{\operatorname{ran}(\mathbf{C})}\left(\mathbf{z}^{(k+1)}-\mathbf{z}^*\right)\\ &=\lim_{k\rightarrow\infty}\prod_{\operatorname{ran}(\mathbf{C})}\mathbf{P}\left(\mathbf{z}^{(k)}-\mathbf{z}^*-2\rho\mathbf{C}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^*\right)\right)\\ &=\lim_{k\rightarrow\infty}\mathbf{P}\prod_{\operatorname{ran}(\mathbf{C})}\mathbf{P}\left(\mathbf{z}^{(k)}-\mathbf{z}^*\right)\\ &=\lim_{k\rightarrow\infty}\prod_{\operatorname{ran}(\mathbf{PC})}\left(\mathbf{z}^{(k)}-\mathbf{z}^*\right),\end{aligned}\tag{3.19}$$

where the second line uses Eq. (3.8), the third line uses that $\lim_{k\rightarrow\infty}\mathbf{x}^{(k+1)}=\mathbf{x}^*$ and that $\mathbf{P}$ is full rank, while the last line exploits that $\mathbf{P}=\mathbf{P}^{-1}$ such that $\mathbf{P}\prod_{\operatorname{ran}(\mathbf{C})}\mathbf{P}=\prod_{\operatorname{ran}(\mathbf{PC})}$. Combining (3.18) and (3.19), finally demonstrates that, under the restrictions of strong convexity and differentiability of f, that

$$\lim_{k\rightarrow\infty}\prod_{\operatorname{ran}(\mathbf{C})+\operatorname{ran}(\mathbf{PC})}\left(\mathbf{z}^{(2k)}-\mathbf{z}^*\right)=\lim_{k\rightarrow\infty}\left(\mathbf{z}^{(2k)}-\mathbf{z}^{\diamond}\right)=\mathbf{0}.$$

Primal convergence follows from Lemma 3.3.1, by noting,

$$\begin{aligned}\mathbf{x}^{(k+1)}&=\left(\nabla f+\rho\mathbf{C}^T\mathbf{C}\right)^{-1}\mathbf{C}^T\left(\mathbf{z}^{(k)}+\rho\mathbf{d}\right)\\ \mathbf{x}^*&=\left(\nabla f+\rho\mathbf{C}^T\mathbf{C}\right)^{-1}\mathbf{C}^T\left(\mathbf{z}^*+\rho\mathbf{d}\right).\end{aligned}\tag{3.20}$$

The equality in this case follows from the fact that ∇f is μ-strongly monotone such that $\left(\nabla f+\rho\mathbf{C}^T\mathbf{C}\right)^{-1}$ is Lipschitz continuous and thus single-valued. Substituting (3.20) into

the primal error, it follows that

$$\begin{aligned}\|\mathbf{x}^{(k+1)}-\mathbf{x}^*\|^2 &= \|\left(\nabla f+\rho\mathbf{C}^T\mathbf{C}\right)^{-1}\mathbf{C}^T\left(\mathbf{z}^{(k)}+\rho\mathbf{d}\right)-\left(\nabla f+\rho\mathbf{C}^T\mathbf{C}\right)^{-1}\mathbf{C}^T\left(\mathbf{z}^*+\rho\mathbf{d}\right)\|^2\\ &\leq\frac{1}{\mu^2}\|\mathbf{C}^T\left(\mathbf{z}^{(k)}-\mathbf{z}^*\right)\|^2\\ &\leq\frac{\sigma^2_{\max}(\mathbf{C})}{\mu^2}\|\mathbf{z}^{(k)}-\mathbf{z}^\diamond\|^2,\end{aligned} \tag{3.21}$$

where, $\sigma_{\max}$ denotes the largest singular value of a matrix.

The primal error $\|\mathbf{x}^{(k+1)}-\mathbf{x}^*\|^2$ is therefore upper bounded by the projected auxiliary error and thus converges.

3.4.4. AVERAGED PDMM CONVERGENCE

As with other operator splitting methods, PDMM can be combined with an averaging stage to guarantee convergence $\forall f\in\Gamma_0$, even those which do not satisfy the strong convexity or differentiability assumptions introduced in Section 3.4.3. The general form of the averaged PDMM operator is given by

$$\mathbf{T}_{P,\rho,\alpha}=(1-\alpha)\mathbf{I}+\alpha\mathbf{T}_{P,\rho},$$

where the scalar $\alpha\in(0,1)$. In the particular case that $\alpha=\frac{1}{2}$, averaged PDMM is equivalent to ADMM, as was previously noted in Section 3.3.6. In this case, by [34, Proposition 4.4], the operator $\mathbf{T}_{P,\rho,\alpha}$ is firmly nonexpansive.

The fixed-point iteration of $\mathbf{T}_{P,\rho,\alpha}$ is therefore given by

$$\mathbf{z}^{(k+1)}=(1-\alpha)\mathbf{z}^{(k)}+\alpha\mathbf{T}_{P,\rho}\mathbf{z}^{(k)}.$$

This is referred to as the α-Krasnosel'skiĭ-Mann iteration [34] of the operator $\mathbf{T}_{P,\rho}$ which is a well documented method of guaranteeing convergence for nonexpansive operators. Notably, recursively applying [34, Eq. 5.16], it follows that the fixed-point residual $\left(\mathbf{T}_{P,\rho}-\mathbf{I}\right)\left(\mathbf{z}^{(k)}\right)$ converges at an asymptotic rate of $\mathcal{O}\left(\frac{1}{k}\right)$ and thus $\mathbf{z}^{(k)}$ converges to a point in $\operatorname{fix}\left(\mathbf{T}_{P,\rho}\right)$ for finite dimensional problems.

3.4.5. LACK OF CONVERGENCE OF PDMM FOR $f\in\Gamma_0$

Without the use of averaging, the convergence results demonstrated so far require f to be both strongly convex and differentiable. While such a result is well known in the case of PR splitting, it is not noted in the existing analysis of PDMM within the literature [50].

In the following, we reinforce the importance of this result by demonstrating a problem instance were PDMM does not converge despite $f\in\Gamma_0$. For this purpose we consider solving the following problem over two nodes.

$$\begin{aligned}\min_{x_1,x_2}\quad & |x_1-1|+|x_2+1|\\ \text{s.t.}\quad & x_1-x_2=0.\end{aligned} \tag{3.22}$$

The objective in (3.22) is neither differentiable nor strongly convex. From Lemmas 3.3.1 and 3.3.2, the primal and auxiliary updates for PDMM are given respectively by

$$\begin{aligned} x_1^{(t+1)} &= \operatorname*{argmin}_x \left(|x-1| - z_{1|2}^{(t)} x + \frac{\rho}{2}\|x\|^2 \right), \\ x_2^{(t+1)} &= \operatorname*{argmin}_x \left(|x+1| + z_{2|1}^{(t)} x + \frac{\rho}{2}\|x\|^2 \right), \\ z_{1|2}^{(t+1)} &= z_{2|1}^{(t)} + 2\rho x_2^{(t+1)}, \; z_{2|1}^{(t+1)} = z_{1|2}^{(t)} - 2\rho x_1^{(t+1)}, \end{aligned} \tag{3.23}$$

By setting $z_{1|2}^{(0)} = z_{2|1}^{(0)} = 0$ and $\rho = 1$ it follows from (3.23) that after the first iteration $x_1^{(1)} = -x_2^{(1)} = 1$ and $z_{1|2}^{(1)} = z_{2|1}^{(1)} = 2$. Note that $x_1 \neq x_2$ such that $\mathbf{x}$ is not primal feasible.

For the second iteration $x_1^{(2)} = -x_2^{(2)} = -1$ and $z_{1|2}^{(2)} = z_{2|1}^{(2)} = 0$. Again, $x_1 \neq x_2$ and furthermore the auxiliary variables are back to their original configuration. The auxiliary variables of PDMM are therefore stuck in a limit cycle and can never converge for this problem. The primal variables also exhibit a limit cycle in this case. As such, $f \in \Gamma_0$ is not a sufficient condition for the convergence of PDMM without the use of operator averaging.

3.5. GEOMETRIC CONVERGENCE

While PR splitting is well known to converge geometrically under the assumption of strong monotonicity and Lipschitz continuity, such conditions cannot be guaranteed in the case of PDMM due to the row rank deficiency of $\mathbf{C}$. However, by assuming that f is strongly convex and has a Lipschitz continuous gradient, we can demonstrate a geometrically contracting upper bound for the primal error of PDMM despite this fact.

3.5.1. A PRIMAL GEOMETRIC CONVERGENCE BOUND FOR STRONGLY CONVEX AND SMOOTH FUNCTIONS

In the following we demonstrate that for strongly convex functions with Lipschitz continuous gradients, the primal variables of PDMM converge at a geometric rate. More formally we show that $\exists\, \epsilon \geq 0, \gamma \in [0,1)$ so that

$$\forall k \in \mathbb{N},\ \|\mathbf{x}^{(k)} - \mathbf{x}^*\|^2 \leq \gamma^k \epsilon.$$

As in the case of Section 3.4.1, this is achieved by firstly forming a geometric bound for the projected auxiliary error

$$\| \underset{\text{ran}(\mathbf{C})+\text{ran}(\mathbf{PC})}{\Pi} \left(\mathbf{z}^{(k)} - \mathbf{z}^* \right) \|^2 = \|\mathbf{z}^{(k)} - \mathbf{z}^\diamond\|^2,$$

before linking back to the primal variables.

The process of bounding the projected auxiliary error is broken down into two stages. Firstly, in Sections 3.5.2 and 3.5.3 we demonstrate how, for strongly convex functions with Lipschitz continuous gradients, PDMM is contractive over a subspace. In Sections 3.5.4 and 3.5.5 we then show how a geometric convergence bound can be found by linking PDMM with a generalized form of the alternating method of projections allowing us to derive the aforementioned γ and ϵ.

3.5.2. CONTRACTIVE NATURE OF PDMM OVER A SUBSPACE

Proving that the projected auxiliary error of PDMM converges geometrically relies on strong monotonicity and the additional notion of Lipschitz continuity. This is defined as follows.

Definition 3.5.1. *Lipschitz Continuous: An operator* $\mathbf{T}: \mathscr{X} \to \mathscr{Y}$ *is* L*-Lipschitz if*

$$\forall\, (\mathbf{x},\mathbf{u}), (\mathbf{y},\mathbf{v}) \in \operatorname{gra}(\mathbf{T}) : \|\mathbf{u}-\mathbf{v}\| \leq L\|\mathbf{x}-\mathbf{y}\|.$$

If $L=1$, $\mathbf{T}$ *is nonexpansive while if* $L<1$ *it is contractive.*

Given this notion, we demonstrate the contractive nature of the PDMM operator over $\operatorname{ran}(\mathbf{C})$ by showing that $\mathbf{C}\nabla f^*\mathbf{C}^T$ is strongly monotone and Lipschitz continuous over this subspace. This is summarized in Lemma 3.5.1.

Lemma 3.5.1. *If* f *is* μ*-strongly convex and* ∇f *is* β*-Lipschitz continuous then* $\mathbf{C}\nabla f^*\mathbf{C}^T$ *is*

(i) $\frac{\sigma^2_{\max}(\mathbf{C})}{\mu}$*-Lipschitz continuous*

(ii) $\frac{\sigma^2_{\min\neq 0}(\mathbf{C})}{\beta}$*-strongly monotone* $\forall \mathbf{z} \in \operatorname{ran}(\mathbf{C})$,

where $\sigma_{\min\neq 0}$ *denotes the smallest non-zero singular value.*

The proof of this lemma can be found in Appendix 3.D. Lemma 3.5.1 reflects a similar approach in [82] for general PR splitting problems. Note that the result demonstrated therein does not hold in this context due to the row-rank deficiency of $\mathbf{C}$. Specifically, [82, Assumption 2] is violated.

As $\mathbf{C}\nabla f^*(\mathbf{C}^T\bullet)$ is both strongly monotone and Lipschitz continuous over $\operatorname{ran}(\mathbf{C})$, from [82], $\mathbf{R}_{\mathbf{T}_1,\rho}$ is contractive $\forall \mathbf{z} \in \operatorname{ran}(\mathbf{C})$ with an upper bound on this contraction given by

$$\delta = \max\left(\frac{\rho\frac{\sigma^2_{\max}(\mathbf{C})}{\mu}-1}{\rho\frac{\sigma^2_{\max}(\mathbf{C})}{\mu}+1}, \frac{1-\rho\frac{\sigma^2_{\min\neq 0}(\mathbf{C})}{\beta}}{1+\rho\frac{\sigma^2_{\min\neq 0}(\mathbf{C})}{\beta}}\right) \in [0,1).$$

By the same arguments, the operator $\mathbf{P}\circ\mathbf{R}_{\mathbf{T}_1,\rho}\circ\mathbf{P}$ is δ contractive over $\operatorname{ran}(\mathbf{PC})$. Using the definition of the PDMM operator (3.7), the two-step PDMM updates given in (3.14), can equivalently be written as

$$\mathbf{z}^{(k+2)} = \left(\mathbf{P}\circ\mathbf{R}_{\mathbf{T}_1,\rho}\circ\mathbf{P}\right)\circ\mathbf{R}_{\mathbf{T}_1,\rho}\left(\mathbf{z}^{(k)}\right).$$

Every two PDMM iterations is therefore the composition of the operators $\mathbf{R}_{\mathbf{T}_1,\rho}$ and $\mathbf{P}\circ\mathbf{R}_{\mathbf{T}_1,\rho}\circ\mathbf{P}$ with each being δ-contractive over $\operatorname{ran}(\mathbf{C})$ and $\operatorname{ran}(\mathbf{PC})$ respectively.

3.5.3. INEQUALITIES DUE TO THE CONTRACTION OF PDMM

The contractive nature of $\mathbf{R}_{\mathbf{T}_1,\rho}$ and $\mathbf{P}\circ\mathbf{R}_{\mathbf{T}_1,\rho}\circ\mathbf{P}$ leads to two important inequalities. In this case we will assume that k is even and that $\mathbf{z}^{\diamond}$ is defined as per (3.16).

Beginning with the operator $\mathbf{R}_{\mathbf{T}_1,\rho}$, consider the updates $\mathbf{y}^\diamond = \mathbf{R}_{\mathbf{T}_1,\rho}(\mathbf{z}^\diamond)$ and $\mathbf{y}^{(k+1)} = \mathbf{R}_{\mathbf{T}_1,\rho}(\mathbf{z}^{(k)})$. Using Lemma 3.3.1, it follows that

$$\begin{aligned}\mathbf{y}^{(k+1)} - \mathbf{y}^\diamond =& 2\boldsymbol{\lambda}^{(k+1)} - \mathbf{z}^{(k)} - \left(2\boldsymbol{\lambda}^\diamond - \mathbf{z}^\diamond\right)\\ =& \mathbf{z}^{(k)} - \mathbf{z}^\diamond - 2\rho\mathbf{C}\left(\mathbf{x}^{(k+1)} - \mathbf{x}^*\right),\end{aligned}$$

so that the projection onto $\ker(\mathbf{C}^T)$ satisfies

$$\prod_{\ker(\mathbf{C}^T)}\left(\mathbf{y}^{(k+1)} - \mathbf{y}^\diamond\right) = \prod_{\ker(\mathbf{C}^T)}\left(\mathbf{z}^{(k)} - \mathbf{z}^\diamond\right).$$

Combining with the δ-contractive nature of $\mathbf{R}_{\mathbf{T}_1,\rho}$ over $\mathrm{ran}(\mathbf{C})$, it follows that,

$$\begin{aligned}\|\mathbf{y}^{(k+1)} - \mathbf{y}^\diamond\|^2 \leq \delta^2 \|& \prod_{\mathrm{ran}(\mathbf{C})}\left(\mathbf{z}^{(k)} - \mathbf{z}^\diamond\right)\|^2\\ &+ \|\prod_{\ker(\mathbf{C}^T)}\left(\mathbf{z}^{(k)} - \mathbf{z}^\diamond\right)\|^2.\end{aligned}$$

For the operator $\mathbf{P}\circ\mathbf{R}_{\mathbf{T}_1,\rho}\circ\mathbf{P}$, as $\mathbf{z}^\diamond = \mathbf{P}\circ\mathbf{R}_{\mathbf{T}_1,\rho}\circ\mathbf{P}\left(\mathbf{y}^\diamond\right)$ by the results of Section 3.4.2 and $\mathbf{z}^{(k+2)} = \mathbf{P}\circ\mathbf{R}_{\mathbf{T}_1,\rho}\circ\mathbf{P}\left(\mathbf{y}^{(k+1)}\right)$, it can be similarly shown that

$$\prod_{\ker(\mathbf{C}^T\mathbf{P})}\left(\mathbf{z}^{(k+2)} - \mathbf{z}^\diamond\right) = \prod_{\ker(\mathbf{C}^T\mathbf{P})}\left(\mathbf{y}^{(k+1)} - \mathbf{y}^\diamond\right),$$

and furthermore that

$$\begin{aligned}\|\mathbf{z}^{(k+2)} - \mathbf{z}^\diamond\|^2 \leq \delta^2 \|& \prod_{\mathrm{ran}(\mathbf{PC})}\left(\mathbf{y}^{(k+1)} - \mathbf{y}^\diamond\right)\|^2\\ &+ \|\prod_{\ker(\mathbf{C}^T\mathbf{P})}\left(\mathbf{y}^{(k+1)} - \mathbf{y}^\diamond\right)\|^2.\end{aligned}$$

While the contractive nature of $\mathbf{R}_{\mathbf{T}_1,\rho}$ and $\mathbf{P}\circ\mathbf{R}_{\mathbf{T}_1,\rho}\circ\mathbf{P}$ suggests the geometric convergence of PDMM, it is unclear what this convergence rate may be. In the following, this will be addressed by deriving a geometric error bound for two-step PDMM by connecting it with the method of alternating projections.

3.5.4. A GEOMETRIC RATE BOUND FOR PDMM INTERPRETED AS AN OPTIMIZATION PROBLEM

Using the results of Section 3.5.3 we now demonstrate that there exists a γ so that the projected auxiliary error satisfies

$$\|\mathbf{z}^{(k+2)} - \mathbf{z}^\diamond\|^2 \leq \gamma^2\|\mathbf{z}^{(k)} - \mathbf{z}^\diamond\|^2, \tag{3.24}$$

where γ^2 can be computed via a non-convex optimization problem. Specifically, it is the maximum objective value of

$$\max_{\mathbf{y},\mathbf{z},\hat{\mathbf{z}}} \quad \|\hat{\mathbf{z}} - \mathbf{z}^\diamond\|^2 \tag{3.25a}$$

$$\text{s.t.} \quad \mathbf{y} = \mathbf{R}_{\mathbf{T}_1,\rho}(\mathbf{z}) \tag{3.25b}$$

$$\hat{\mathbf{z}} = \mathbf{P}\circ\mathbf{R}_{\mathbf{T}_1,\rho}\circ\mathbf{P}\left(\mathbf{y}\right) \tag{3.25c}$$

$$\|\mathbf{z} - \mathbf{z}^\diamond\|^2 \leq 1. \tag{3.25d}$$

Here, (3.25a) captures the worst case improvement in the distance between the two-step iterates ($\hat{\mathbf{z}}$) and the projected fixed point ($\mathbf{z}^\diamond$). Due to (3.25d), the maximum of this objective exactly determines the worst case convergence rate. The vector $\mathbf{z}$ corresponds to the initial auxiliary variable, $\mathbf{y}$ and $\hat{\mathbf{z}}$ are generated via the one and two step PDMM updates imposed by (3.25b) and (3.25c), and (3.25d) defines the feasible set of $\mathbf{z}$. In a similar manner to (3.16), $\mathbf{z}^\diamond = \mathbf{z}^* + \underset{\ker(\mathbf{C}^T)\cap\ker(\mathbf{C}^T\mathbf{P})}{\Pi}(\mathbf{z})$ so that $\mathbf{z}-\mathbf{z}^\diamond \in \operatorname{ran}(\mathbf{PC}) + \operatorname{ran}(\mathbf{C})$.

Using the properties of $\mathbf{R}_{\mathbf{T}_1,\rho}$ and $\mathbf{P}\circ\mathbf{R}_{\mathbf{T}_1,\rho}\circ\mathbf{P}$ from Section 3.5.3, the optimum of (3.25) can be equivalently computed via

$$\begin{aligned}
\max_{\mathbf{y},\mathbf{z}} \quad & \left\| \left(\delta \underset{\operatorname{ran}(\mathbf{PC})}{\Pi} + \underset{\ker(\mathbf{C}^T\mathbf{P})}{\Pi} \right) (\mathbf{y}-\mathbf{y}^\diamond) \right\|^2 \\
\text{s.t.} \quad & \left\| \underset{\operatorname{ran}(\mathbf{C})}{\Pi} (\mathbf{y}-\mathbf{y}^\diamond) \right\|^2 \le \delta^2 \left\| \underset{\operatorname{ran}(\mathbf{C})}{\Pi} (\mathbf{z}-\mathbf{z}^\diamond) \right\|^2 && \text{(3.26a)} \\
& \underset{\ker(\mathbf{C}^T)}{\Pi} (\mathbf{y}-\mathbf{y}^\diamond) = \underset{\ker(\mathbf{C}^T)}{\Pi} (\mathbf{z}-\mathbf{z}^\diamond) && \text{(3.26b)} \\
& \|\mathbf{z}-\mathbf{z}^\diamond\|^2 \le 1, && \text{(3.26c)}
\end{aligned}$$

where $\mathbf{y}^\diamond = \mathbf{R}_{\mathbf{T}_1,\rho}(\mathbf{z}^\diamond)$ and in the objective we have exploited the orthogonality of $\operatorname{ran}(\mathbf{PC})$ and $\ker(\mathbf{C}^T\mathbf{P})$. The constraints of (3.26) increase the feasible sets of $\mathbf{y}$ and $\hat{\mathbf{z}}$ while including the true updates due to $\mathbf{R}_{\mathbf{T}_1,\rho}$ as special cases.

The constraints (3.26a), (3.26b) and (3.26c) collectively define the feasible set of the vectors $\mathbf{y}-\mathbf{y}^\diamond$. We can further simplify (3.26) by considering the form of this feasible set. In particular, as (3.26c) denotes a sphere, the constraints (3.26a) and (3.26b) restrict the vectors $\mathbf{y}-\mathbf{y}^\diamond$ to lie in an ellipsoid given by

$$\mathbf{y}-\mathbf{y}^\diamond \in \left\{ \left(\delta \underset{\operatorname{ran}(\mathbf{C})}{\Pi} + \underset{\ker(\mathbf{C}^T)}{\Pi} \right) \mathbf{u} \mid \|\mathbf{u}\| \le 1 \right\}.$$

By defining the additional variable $\mathbf{u} = \mathbf{z}-\mathbf{z}^\diamond$, the optimization problem in (3.25) is therefore equivalent to

$$\begin{aligned}
\max_{\mathbf{u}} \quad & \left\| \left(\delta \underset{\operatorname{ran}(\mathbf{PC})}{\Pi} + \underset{\ker(\mathbf{C}^T\mathbf{P})}{\Pi} \right) \left(\delta \underset{\operatorname{ran}(\mathbf{C})}{\Pi} + \underset{\ker(\mathbf{C}^T)}{\Pi} \right) \mathbf{u} \right\|^2 \\
\text{s.t.} \quad & \|\mathbf{u}\|^2 \le 1, \mathbf{u} \in \operatorname{ran}(\mathbf{PC}) + \operatorname{ran}(\mathbf{C}), && \text{(3.27)}
\end{aligned}$$

where the additional domain constraint stems from the definition of $\mathbf{z}^\diamond$. In the following we demonstrate how (3.27) exhibits an analytic expression for γ, ultimately allowing us to form our primal convergence rate bound.

3.5.5. RELATIONSHIP WITH THE METHOD ALTERNATING OF PROJECTIONS

To compute the contraction factor γ in (3.24), we can exploit the relationship between (3.27) and the method of alternating projections. Optimal rate bounds for generalizations of the classic alternating projections algorithm has been an area of recent attention in the literature with two notable papers on the subject being [88] and [89]. Our analysis below follows in the spirit of these methods.

Consider the particular operator from Eq. (3.27),

$$\mathbb{G} = \left(\delta \underset{\text{ran}(\mathbf{PC})}{\Pi} + \underset{\ker(\mathbf{C}^T\mathbf{P})}{\Pi}\right)\left(\delta \underset{\text{ran}(\mathbf{C})}{\Pi} + \underset{\ker(\mathbf{C}^T)}{\Pi}\right).$$

Given the domain constraint also from (3.27), it follows that γ corresponds to the largest singular value of the matrix $\underset{\text{ran}(\mathbf{C})+\text{ran}(\mathbf{PC})}{\Pi} \mathbb{G}^T\mathbb{G} \underset{\text{ran}(\mathbf{C})+\text{ran}(\mathbf{PC})}{\Pi}$. We can therefore compute γ by taking advantage of the structure of $\mathbb{G}$. In particular, from [89], there exists an orthonormal matrix $\mathbf{D}$ such that

$$\underset{\text{ran}(\mathbf{C})}{\Pi} = \mathbf{D}\begin{bmatrix}\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix}\mathbf{D}^T, \quad \underset{\text{ran}(\mathbf{PC})}{\Pi} = \mathbf{D}\begin{bmatrix}\cos^2(\boldsymbol{\theta}) & \cos(\boldsymbol{\theta})\sin(\boldsymbol{\theta}) & \mathbf{0} & \mathbf{0}\\ \cos(\boldsymbol{\theta})\sin(\boldsymbol{\theta}) & \sin^2(\boldsymbol{\theta}) & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix}\mathbf{D}^T,$$

where $\cos(\boldsymbol{\theta})$ and $\sin(\boldsymbol{\theta})$ denote diagonal matrices of the cosines and sines of the principal angles between ran($\mathbf{C}$) and ran($\mathbf{PC}$), respectively. It follow that for the considered operator

$$\mathbb{G} = \mathbf{D}\begin{bmatrix}\delta^2+\delta(1-\delta)\sin(\boldsymbol{\theta})^2 & -(1-\delta)\cos(\boldsymbol{\theta})\sin(\boldsymbol{\theta})^2 & \mathbf{0} & \mathbf{0}\\ -\delta(1-\delta)\cos(\boldsymbol{\theta})\sin(\boldsymbol{\theta}) & (1-\delta)\cos(\boldsymbol{\theta})^2+\delta & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \delta\mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}\end{bmatrix}\mathbf{D}^H.$$

Note that the bottom right identity matrix corresponds to those vectors that lie outside our feasible set.

Given the structure of $\mathbb{G}$ and the diagonal nature of $\mathscr{C}$ and $\mathscr{S}$, it follows that γ is either given by δ or by $\sigma_{\max}$ of any of the two by two submatrices

$$\mathbb{G}_i = \begin{bmatrix}\delta^2+\delta(1-\delta)\sin(\theta_i)^2 & -(1-\delta)\cos(\theta_i)\sin(\theta_i)\\ -\delta(1-\delta)\cos(\theta_i)\sin(\theta_i) & \delta+(1-\delta)\cos(\theta_i)^2\end{bmatrix},$$

where $\theta_i \in (0, \frac{\pi}{2}]$ is the ith principal angle. The singular values of such a submatrix can be computed via the following lemma.

Lemma 3.5.2. *The singular values of $\mathbb{G}_i$ are given by*

$$\sigma(\mathbb{G}_i) = \sqrt{\delta^2+(1-\delta^2)\cos(\theta_i)\left(\frac{(1-\delta^2)\cos(\theta_i)}{2} \pm \sqrt{\frac{(1-\delta^2)^2\cos(\theta_i)^2}{4}+\delta^2}\right)}.$$

The proof for this lemma can be found in Appendix 3.E. As the singular values are a nondecreasing function of $\cos(\theta_i)$ and thus a nonincreasing function of θ_i, it follows that

$$\gamma = \max\{\delta, \{\sigma_{\max}(\mathbb{G}_i)\ \forall i\}\} = \sigma_{\max}(\mathbb{G}_F). \tag{3.28}$$

Here $\mathbb{G}_F$ refers to the submatrix associated with the smallest non-zero principal angle θ_F, which is referred to as the Friedrichs angle. Therefore, given δ and $\cos(\theta_F)$,

$$\gamma = \sqrt{\delta^2+(1-\delta^2)\cos(\theta_F)\left(\frac{(1-\delta^2)\cos(\theta_F)}{2}+\sqrt{\frac{(1-\delta^2)^2\cos(\theta_F)^2}{4}+\delta^2}\right)}.$$

3.5.6. FROM AN AUXILIARY ERROR BOUND TO A GEOMETRIC PRIMAL CONVERGENCE BOUND

Using (3.28), our primal convergence bound can finally be constructed. For two-step PDMM we already know that

$$\|\mathbf{z}^{(k+2)} - \mathbf{z}^{\diamond}\|^2 \leq \gamma^2 \|\mathbf{z}^{(k)} - \mathbf{z}^{\diamond}\|^2.$$

By recursively applying this result, it follows that, for even k,

$$\begin{aligned}\|\mathbf{z}^{(k+1)} - \mathbf{T}_{P,\rho}(\mathbf{z}^{\diamond})\|^2 \leq & \gamma^k \|\mathbf{z}^1 - \mathbf{T}_{P,\rho}(\mathbf{z}^{\diamond})\|^2 \\ \leq & \gamma^k \|\mathbf{z}^0 - \mathbf{z}^{\diamond}\|^2,\end{aligned}$$

so that the projected auxiliary error of PDMM satisfies

$$\|\mathbf{z}^{(k+2)} - \mathbf{z}^{\diamond}\|^2 \leq \gamma^{k+2} \frac{\|\mathbf{z}^0 - \mathbf{z}^{\diamond}\|^2}{\gamma}. \tag{3.29}$$

By applying (3.21) to (3.29), the final primal bound is given by

$$\begin{aligned}\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\|^2 \leq & \frac{\sigma_{\max}^2(\mathbf{C})}{\mu^2} \|\mathbf{z}^{(k)} - \mathbf{z}^{\diamond}\|^2 \\ \leq & \gamma^{k+2} \frac{\sigma_{\max}^2(\mathbf{C})}{\mu^2 \gamma} \|\mathbf{z}^{(0)} - \mathbf{z}^{\diamond}\|^2\end{aligned} \tag{3.30}$$

The primal error $\|\mathbf{x}^{(k+2)} - \mathbf{x}^*\|^2$ is therefore upper bounded by a geometrically contracting sequence and thus converges at a geometric rate. To the best of the authors knowledge, this is the fastest rate for PDMM proven within the literature.

3.6. NUMERICAL EXPERIMENTS

In this section, we verify the analytical results of Section 3.4 and 3.5 with numerical experiments. These results are broken down into two subsections: the convergence of PDMM for strongly convex and differentiable functions and the geometric convergence of PDMM for strongly convex functions with Lipschitz continuous gradients.

3.6.1. PDMM FOR STRONGLY CONVEX AND DIFFERENTIABLE FUNCTIONS

The first set of simulations validate the sufficiency of strong convexity and differentiability to guarantee primal convergence, as introduced in Section 3.4. For these simulations, as testing all such functions would be computationally infeasible, we instead considered the family of m-th power of m-norms for $m \in \{3,4,5,\cdots\}$ combined with an additive squared Euclidean norm term to enforce strong convexity. The prototype problem for these simulations is given by

$$\begin{aligned}\min_{\mathbf{x}} \quad & \sum_{i \in V} \left(\|\mathbf{x}_i - \mathbf{a}_i\|_m^m + \mu \|\mathbf{x}_i - \mathbf{a}_i\|^2 \right) \\ \text{s.t.} \quad & \mathbf{x}_i - \mathbf{x}_j = \mathbf{0} \quad \forall (i,j) \in E,\end{aligned}$$

where $\mathbf{a}_i$ are local observation vectors, μ controls the strong convexity parameter and, for simplicity, edge based consensus constraints were chosen.

An $N = 10$ node undirected Erdős-Rényi network [90] was considered for these simulations. Such networks are randomly generated graphs where $\forall\ i,\ j \in V \setminus i$, there is equal probability that $(i, j) \in E$. This probability determines the density of the connectivity in the network and in this case was set to $\frac{\log(N)}{N}$. The resulting network had 12 undirected edges and was verified as forming a single connected component as per the assumptions in Section 3.3. Additionally, a randomly generated initial $\mathbf{z}^{(0)}$ was also used for all problem instances. Finally the strong convexity parameter was set to $\mu = 10^{-3}$.

For $m = 3, \cdots 10$, 150 iterations of PDMM were performed and the resulting primal error computed. The squared Euclidean distance between the primal iterates and the primal optimal set was used as an error measure. Figure 3.2 demonstrates the convergence of this error with respect to iteration count. For each m the step sizes ρ were empirically selected to optimize convergence rate. Note that the finite precision stems

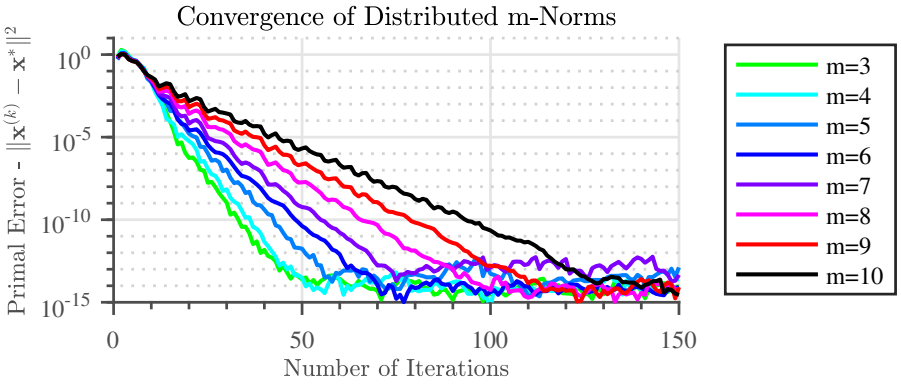


Figure 3.2: The primal convergence of different m-norm^m consensus problem for a 10 node Erdős-Rényi network.

from the use of MATLABs *fminunc* function.

Figure 3.3 further demonstrates that the choice of ρ does not effect the guarantee of convergence which in this instance was modeled via the number of iterations required to reach an auxiliary precision of $1e^{-5}$. This measure was chosen as the auxiliary error is monotonically decreasing with iteration count. In contrast the primal error need not satisfy this point, as can be observed in Figure 3.2. Note that while there is a clear variation in the rate of convergence for different choices of ρ, the guarantee of convergence of the algorithms are unaffected.

3.6.2. GEOMETRIC CONVERGENCE OF PDMM FOR STRONGLY CONVEX AND SMOOTH FUNCTIONS

The final simulations verify the geometric bound from Section 3.5 by comparing the convergence of multiple problem instances to (3.30). Specifically, 10^4 random quadratic op-

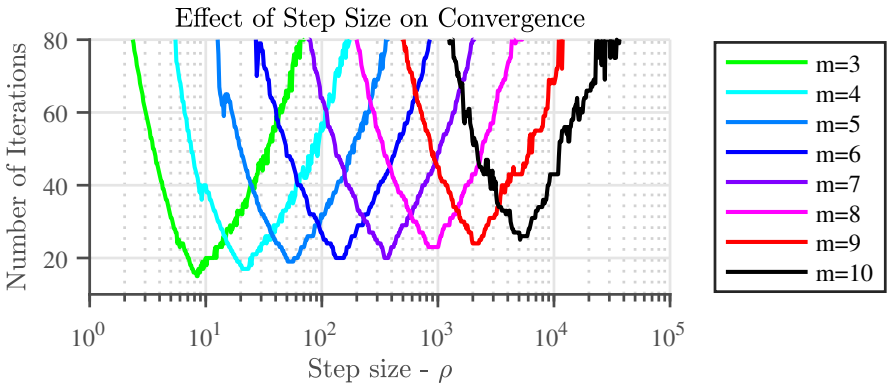


Figure 3.3: A comparison to the required iterations for $\|\mathbf{z}^{(k)} - \mathbf{z}^{\diamond}\|^2 \leq 1e^{-5}$ for various step sizes (ρ). The step size is plotted on a log scale to better demonstrate the convergence characteristics of the different problems.

timisation problems were generated, each of the form

$$\begin{aligned} \min_{\mathbf{x}} \quad & \sum_{i \in V} \left(\frac{1}{2} \mathbf{x}_i^T \mathbf{Q}_i \mathbf{x}_i - \mathbf{q}_i^T \mathbf{x}_i \right) \\ \text{s.t.} \quad & \mathbf{x}_i - \mathbf{x}_j = \mathbf{0} \quad \forall (i,j) \in E. \end{aligned}$$

For each problem, the local variables were configured so that $\mathbf{x}_i \in \mathbb{R}^3 \ \forall i \in V$ and the resulting objective was paired with a random 10 node Erdős-Rényi network. The connection probability of each network was set to $\frac{\log(N)}{N}$ and the networks were verified as forming single connected components.

For each problem instance, the matrices $\mathbf{Q}_i \succeq \mathbf{0}$ were generated in such a way that a constant convergence rate bound was achieved. In this case the contraction factor of this rate bound was specified as $\gamma = 0.9$. Furthermore, the initial vector $\mathbf{z}^{(0)}$ was generated randomly and for each the associated $\mathbf{z}^{\diamond}$ was computed as per Eq. (3.16). This randomization procedure was implemented so that $\frac{\sigma_{\max}^2(\mathbf{C})}{\mu^2 \gamma} \|\mathbf{z}^{(0)} - \mathbf{z}^{\diamond}\|^2 = 1$ for all instances.

For each problem instance, a total of 120 iterations of PDMM, were performed and the auxiliary errors, $\|\mathbf{z}^{(k)} - \mathbf{z}^{\diamond}\|^2$ for k even and $\|\mathbf{z}^{(k)} - \mathbf{T}_{P,\rho}(\mathbf{z}^{\diamond})\|^2$ for k odd, were computed. The distribution of the resulting data is demonstrated in Figure 3.4 which highlights the spread of the convergence curves across all problem instances.

As expected, (3.29) provides a strict upper bound for all problem instances, with the smoothness of the curves stemming from the linear nature of the PDMM update equations. Furthermore, the rate of the worst case sequence (100% quantile) does not exceed that of the bound. Interestingly, while (3.29) holds for the worst case functions, most problem instances exhibit far faster convergence. This suggests that, for more restrictive problem classes, stronger bounds may exist.

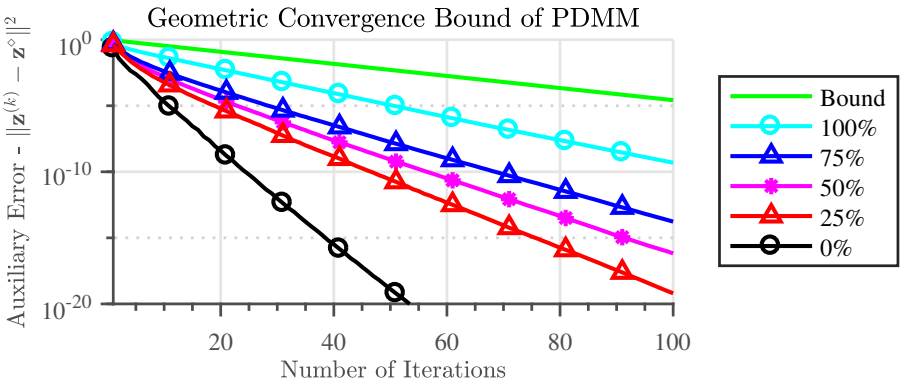


Figure 3.4: Convergence of simulated PDMM problem instances. From top to bottom, the solid green line denotes the convergence rate bound while the remaining 5 lines denote the 100%, 75%, 50%, 25% and 0% quantiles respectively.

3.7. CONCLUSIONS

In this chapter we have presented a novel derivation of the node-based distributed algorithm termed the primal-dual method of multipliers (PDMM). Unlike existing efforts within the literature, monotone operator theory was used for this purpose, providing both a succinct derivation for PDMM while highlighting the relationship between it and other existing first order methods such as PR splitting and ADMM. Using this derivation, primal convergence was demonstrated for strongly convex, differentiable functions and, in the case of strongly convex functions with Lipschitz continuous gradients, a geometric primal convergence bound was presented. This is despite the loss of a full row-rank assumption required by existing approaches and is a first for PDMM. In conclusion, the demonstrated results unify PDMM with existing solvers in the literature while providing new insight into its operation and convergence characteristics.

APPENDICES

3.A. PROOF OF LEMMA 3.3.1

As $\mathbf{R}_{\mathbf{T}_1,\rho} = 2\mathbf{J}_{\mathbf{T}_1,\rho} - \mathbf{I}$, we begin by defining a method for computing the update $\boldsymbol{\lambda}^{(k+1)} = \mathbf{J}_{\mathbf{T}_1,\rho}\left(\mathbf{z}^{(k)}\right)$. Firstly, by the definition of the resolvent,

$$\begin{aligned}\boldsymbol{\lambda}^{(k+1)} &= \left(\mathbf{I} + \rho\mathbf{T}_1\right)^{-1}\left(\mathbf{z}^{(k)}\right)\\ \boldsymbol{\lambda}^{(k+1)} &\in \mathbf{z}^{(k)} - \rho\mathbf{T}_1\left(\boldsymbol{\lambda}^{(k+1)}\right).\end{aligned}$$

From the definition of the operator $\mathbf{T}_1$, it follows that

$$\boldsymbol{\lambda}^{(k+1)} \in \mathbf{z}^{(k)} - \rho\left(\mathbf{C}\partial f^*\left(\mathbf{C}^T\boldsymbol{\lambda}^{(k+1)}\right) - \mathbf{d}\right).$$

Let $\mathbf{x} \in \partial f^*\left(\mathbf{C}^T\boldsymbol{\lambda}\right)$. For $f \in \Gamma_0$, it follows from Proposition 16.10 [34], that $\mathbf{x} \in \partial f^*\left(\mathbf{C}^T\boldsymbol{\lambda}\right) \iff \partial f(\mathbf{x}) \ni \mathbf{C}^T\boldsymbol{\lambda}$ so that

$$\begin{aligned}\boldsymbol{\lambda}^{(k+1)} &= \mathbf{z}^{(k)} - \rho\left(\mathbf{C}\mathbf{x}^{(k+1)} - \mathbf{d}\right)\\ \mathbf{C}^T\boldsymbol{\lambda}^{(k+1)} &\in \partial f\left(\mathbf{x}^{(k+1)}\right).\end{aligned} \tag{3.31}$$

Thus, $\mathbf{x}^{(k+1)}$ can be computed as

$$\mathbf{x}^{(k+1)} = \arg\min_{\mathbf{x}}\left(f(\mathbf{x}) - \langle\mathbf{C}^T\mathbf{z}^{(k)}, \mathbf{x}\rangle + \frac{\rho}{2}||\mathbf{C}\mathbf{x} - \mathbf{d}||^2\right)$$

Combining (3.31) with the fact that $\mathbf{y}^{(k+1)} = \left(2\mathbf{J}_{\mathbf{T}_1,\rho} - \mathbf{I}\right)\left(\mathbf{z}^{(k)}\right)$ completes the proof. □

3.B. PROOF OF LEMMA 3.3.2

As $\mathbf{R}_{\mathbf{T}_2,\rho} = 2\mathbf{J}_{\mathbf{T}_2,\rho} - \mathbf{I}$, we again begin by defining a method for computing the update $\mathbf{J}_{\mathbf{T}_2,\rho}\left(\mathbf{y}^{(k+1)}\right)$,

From [84, Eq. 1.3], the resolvent of $\iota_{\ker(\mathbf{I}-\mathbf{P})}$, is given by

$$\mathbf{J}_{\mathbf{T}_2,\rho}\left(\mathbf{y}^{(k+1)}\right) = \prod_{\ker(\mathbf{I}-\mathbf{P})}\mathbf{y}^{(k+1)}.$$

It follows that the reflected resolvent can be computed as

$$\mathbf{z}^{(k+1)} = \left(2\prod_{\ker(\mathbf{I}-\mathbf{P})} - \mathbf{I}\right)\mathbf{y}^{(k+1)} = \mathbf{P}\mathbf{y}^{(k+1)},$$

completing the proof. □

3.C. PROOF OF LEMMA 3.4.1

Reconsider the auxiliary PDMM updates given in Eq. (3.8). Substituting (3.8) into (3.15), it follows that

$$\begin{aligned}\|\mathbf{z}^{(k+1)}-\mathbf{z}^*\|^2 &= \|\mathbf{P}\left(\mathbf{z}^{(k)}-\mathbf{z}^*-2\rho\mathbf{C}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^*\right)\right)\|^2\\ &= \|\mathbf{z}^{(k)}-\mathbf{z}^*-2\rho\mathbf{C}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^*\right)\|^2\\ &= \|\mathbf{z}^{(k)}-\mathbf{z}^*\|^2-4\rho\left\langle\boldsymbol{\lambda}^{(k+1)}-\boldsymbol{\lambda}^*,\mathbf{C}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^*\right)\right\rangle\\ &\leq \|\mathbf{z}^{(k)}-\mathbf{z}^*\|^2,\end{aligned} \tag{3.32}$$

where the penultimate line uses the dual update in Lemma 3.3.1 and the final line uses the nonexpansiveness of $\mathbf{T}_{P,\rho}$.

As $\mathbf{C}^T\boldsymbol{\lambda}=\nabla f(\mathbf{x})$ (3.31), by Definition 3.4.3 it follows that,

$$\begin{aligned}&\langle\boldsymbol{\lambda}_1-\boldsymbol{\lambda}_2,\mathbf{C}(\mathbf{x}_1-\mathbf{x}_2)\rangle\geq\mu\|\mathbf{x}_1-\mathbf{x}_2\|^2\\ &\forall\mathbf{x}_1\neq\mathbf{x}_2,\ \underset{\text{ran}(\mathbf{C})}{\Pi}(\boldsymbol{\lambda}_1)\neq\underset{\text{ran}(\mathbf{C})}{\Pi}(\boldsymbol{\lambda}_2).\end{aligned} \tag{3.33}$$

Recursively applying (3.32) and by using (3.33), it follows that

$$\lim_{k\to\infty}4\rho\sum_{i=1}^{k}\mu\|\mathbf{x}^{(i)}-\mathbf{x}^*\|^2\leq\|\mathbf{z}^{(0)}-\mathbf{z}^*\|^2-\lim_{k\to\infty}\|\mathbf{z}^{(k)}-\mathbf{z}^*\|^2$$

so that $\left(\|\mathbf{x}^{(k)}-\mathbf{x}^*\|^2\right)_{k\in\mathbb{N}}$ is finitely summable. If $\left(\|\mathbf{x}^{(k)}-\mathbf{x}^*\|^2\right)_{k\in\mathbb{N}}$ is non-zero infinitely often then $\lim_{k\to\infty}\|\mathbf{x}^{(k)}-\mathbf{x}^*\|^2=0$ and thus $\lim_{k\to\infty}\mathbf{x}^{(k)}=\mathbf{x}^*$.

To demonstrate this point note that if $\exists k\mid\mathbf{x}^{(k+2)}=\mathbf{x}^{(k+1)}=\mathbf{x}^*$ then by the two-step PDMM update given in (3.14), $\mathbf{z}^{(k+2)}=\mathbf{z}^{(k)}$. Thus, $\forall M\geq 1$ the same primal updates will be computed so that $\mathbf{x}^{(k+M)}=\mathbf{x}^{(k+M-1)}=\mathbf{x}^*$.

Any $\mathbf{z}^{(k)}$ which produces two successive primal optimal updates therefore guarantees primal convergence. Thus, given our assumptions on f, any sequence which does not guarantee primal convergence in finite iterations has to be non-zero infinitely often so that $\lim_{k\to\infty}\mathbf{x}^{(k)}=\mathbf{x}^*$. As ∇f is single-valued, it also follows that $\lim_{k\to\infty}\underset{\text{ran}(\mathbf{C})}{\Pi}\left(\boldsymbol{\lambda}^{(k)}\right)=\underset{\text{ran}(\mathbf{C})}{\Pi}(\boldsymbol{\lambda}^*)$. □

3.D. PROOF OF LEMMA 3.5.1

Under the assumption that $f\in\Gamma_0$ is μ-strongly convex and ∇f is β-Lipschitz, from Theorem 18.15 [34], f^* is both $\frac{1}{\beta}$-strongly convex and $\frac{1}{\mu}$-smooth. It follows that ∇f^* is both $\frac{1}{\beta}$ strongly monotone and $\frac{1}{\mu}$ Lipschitz continuous.

In the case of *(i)*, due to the Lipschitz continuity of ∇f^*

$$\begin{aligned}||\mathbf{C}\left(\nabla f^*(\mathbf{C}^T\mathbf{z}_1)-\nabla f^*(\mathbf{C}^T\mathbf{z}_2)\right)||&\leq\sigma_{\max}(\mathbf{C})\,||\nabla f^*(\mathbf{C}^T\mathbf{z}_1)-\nabla f^*(\mathbf{C}^T\mathbf{z}_2)||\\ &\leq\frac{\sigma_{\max}(\mathbf{C})}{\mu}||\mathbf{C}^T(\mathbf{z}_1-\mathbf{z}_2)||\\ &\leq\frac{\sigma^2_{\max}(\mathbf{C})}{\mu}||\mathbf{z}_1-\mathbf{z}_2||,\end{aligned}$$

Therefore, $\mathbf{C}\nabla f^*(\mathbf{C}^T\bullet)$ is $\frac{\sigma_{\max}(\mathbf{C})^2}{\mu}$-Lipschitz continuous. In the case of *(ii)*, due to the strong monotonicity of ∇f^*

$$\langle \mathbf{C}\big(\nabla f^*(\mathbf{C}^T\mathbf{z}_1)-\nabla f^*(\mathbf{C}^T\mathbf{z}_2)\big),\mathbf{z}_1-\mathbf{z}_2\rangle \geq \frac{||\mathbf{C}^T(\mathbf{z}_1-\mathbf{z}_2)||^2}{\beta}.$$

For all $\mathbf{z}_1,\mathbf{z}_2 \in \text{ran}(\mathbf{C})$ it follows that

$$\frac{||\mathbf{C}^T(\mathbf{z}_1-\mathbf{z}_2)||^2}{\beta} \geq \frac{\sigma^2_{\min\neq 0}(\mathbf{C})\,||\mathbf{z}_1-\mathbf{z}_2||^2}{\beta},$$

completing the proof. □

3.E. PROOF OF LEMMA 3.5.2

Consider the two by two matrix

$$\mathbb{G}_i = \begin{bmatrix} \delta^2+\delta(1-\delta)\sin(\theta_i)^2 & -(1-\delta)\cos(\theta_i)\sin(\theta_i) \\ -\delta(1-\delta)\cos(\theta_i)\sin(\theta_i) & \delta+(1-\delta)\cos(\theta_i)^2. \end{bmatrix}$$

The squared singular values of this matrix are given by the eigenvalues of the matrix

$$\mathbb{G}_i^T\mathbb{G}_i = \begin{bmatrix} \delta^4+\delta^2(1-\delta^2)\sin(\theta_i)^2 & -\delta(1-\delta^2)\cos(\theta_i)\sin(\theta_i) \\ -\delta(1-\delta^2)\cos(\theta_i)\sin(\theta_i) & \delta^2+(1-\delta^2)\cos(\theta_i)^2. \end{bmatrix} \tag{3.34}$$

The eigenvalues of (3.34) can be computed via its trace and determinant. With some manipulation, these are given by

$$\text{tr}\left(\mathbb{G}_i^T\mathbb{G}_i\right) = 2\delta^2+(1-\delta^2)^2\cos(\theta_i)^2,\ \det\left(\mathbb{G}_i^T\mathbb{G}_i\right)=\delta^4.$$

It follows that the squared singular values of $\mathbb{G}_i$ are given by

$$\begin{aligned}\sigma^2(\mathbb{G}_i) &= \frac{\text{tr}\left(\mathbb{G}_i^T\mathbb{G}_i\right)}{2} \pm \sqrt{\frac{\text{tr}\left(\mathbb{G}_i^T\mathbb{G}_i\right)^2}{4}-\det\left(\mathbb{G}_i^T\mathbb{G}_i\right)} \\ &= \delta^2+(1-\delta^2)\cos(\theta_i)\left(\frac{(1-\delta^2)\cos(\theta_i)}{2} \pm \sqrt{\frac{(1-\delta^2)^2\cos(\theta_i)^2}{4}+\delta^2}\right),\end{aligned}$$

completing the proof. □

4

Guaranteeing the Convergence of PDMM via Primal Regularization

Thomas Sherson

"The more that you read, the more things you will know.
The more that you learn, the more places you'll go."

Dr. Seuss

As the primal dual method of multipliers (PDMM) cannot be guaranteed to converge for all closed, convex and proper functions, in this chapter we present a modified PDMM algorithm to address this point. Specifically, we demonstrate how by introducing an additional regularization term into the computation of the primal variables we can guarantee that they converge to an optimal state. Furthermore, we show how this modification can be motivated from the perspective of monotone operator theory which again provides the basis for our convergence analysis. This chapter is therefore best viewed as a supplement to Chapter 3 by providing a means to improves the generality of the problems to which PDMM can be applied.

4.1. ORGANIZATION OF THE CHAPTER

This chapter is arranged as follows. In Section 4.2 we introduce appropriate nomenclature to support the remainder of the text. Section 4.3 introduces a modified version of the primal dual method of multipliers based on a primal-dual reformulation. In Section 4.4, we then demonstrate the sufficiency of this modification to guarantee the convergence of the primal variables to an optimal state. Section 4.5 then verifies this convergence in the case of a distributed L1 optimization problem which is also know to not be guaranteed to converge for the standard PDMM algorithm. Finally, in Section 4.6 we draw our conclusions for the chapter.

4.2. NOMENCLATURE

In this work we denote by $\mathbb{R}$ the set of real numbers, by $\mathbb{R}^N$ the set of real column vectors of length N and by $\mathbb{R}^{M\times N}$ the set of M by N real matrices. Let $\mathscr{X},\mathscr{Y}\subseteq\mathbb{R}^N$. A set valued operator $\mathbf{T}:\mathscr{X}\rightarrow\mathscr{Y}$ is defined by its graph, $\text{gra}\,(\mathbf{T})=\{(\mathbf{x},\mathbf{y})\in\mathscr{X}\times\mathscr{Y}\mid\mathbf{y}\in\mathbf{T}(\mathbf{x})\}$. Similarly, the notion of an inverse of an operator $\mathbf{T}^{-1}$ is defined via its graph so that $\text{gra}\left(\mathbf{T}^{-1}\right)=\{(\mathbf{y},\mathbf{x})\in\mathscr{Y}\times\mathscr{X}\mid\mathbf{y}\in\mathbf{T}(\mathbf{x})\}$. $\mathbf{J}_{\mathbf{T},\rho}=\left(\mathbf{I}+\rho\mathbf{T}\right)^{-1}$ denotes the resolvent of an operator while $\mathbf{R}_{\mathbf{T},\rho}=2\mathbf{J}_{\mathbf{T},\rho}-\mathbf{I}$ denotes the reflected resolvent (Cayley operator).

4.3. MODIFYING THE PDMM ALGORITHM

The aim of this section is to devise a simple modification to the PDMM algorithm to address its lack of convergence for general closed, convex and proper functions. Specifically, we want to introduce a modification that doesn't impact on the distributability of the method, i.e. one that does not introduce any additional sharing of information between nodes. The obvious area to incorporate such a modification is in the computation of the primal variables $\mathbf{x}^{(k+1)}$. For the sake of completeness, an equivalent version of the PDMM algorithm, originally proposed in Chapter 3 is provided in Algorithm 3. This equivalent form of the algorithm is adopted as it more naturally facilitates the analysis undertaken in the remainder of this chapter.

An interesting point we can note about the standard PDMM algorithm is that after each dual update $\boldsymbol{\lambda}_{i|j}^{(k+1)}$, all information regarding the primal variables $\mathbf{x}_i^{(k+1)}$ is essentially discarded. However, these primal iterates represent an estimate of the optimal primal variables at each node based on the current information available to them at each iteration. One can therefore think that adding some additional form of primal regularization which penalizes deviation between primal iterates may aid in achieving convergence by exploiting this information. In this work, we consider the use of a standard squared Euclidean norm of the form $\|\mathbf{x}-\mathbf{x}^{(k)}\|^2$ for this regularization. Such a modification is also attractive as it is inherently local to each node and as such introduces no additional communication between nodes. As we show in the later portion of this chapter, this concept not only proves to be sufficient to guarantee primal optimality but can also be derived from monotone operator theory as well.

Algorithm 3 Distributed PDMM

Initialize: $\mathbf{z}^{(0)} \in \mathbb{R}^{M_E}$
for k=0,..., **do**
  for all $i \in V$ **do** ▷ Primal Update
    $\mathbf{x}_i^{(k+1)} = \underset{\mathbf{x}_i}{\operatorname{argmin}} \left(f_i(\mathbf{x}_i) + \sum_{j \in \mathcal{N}(i)} \left(-\left\langle \mathbf{A}_{i|j}^T \mathbf{z}_{i|j}^{(k)}, \mathbf{x}_i \right\rangle + \frac{\rho}{2} \left\| \mathbf{A}_{i|j}\mathbf{x}_i - \frac{\mathbf{b}_{ij}}{2} \right\|^2 \right) \right)$
  for all $j \in \mathcal{N}(i)$ **do** ▷ Dual Update
    $\boldsymbol{\lambda}_{i|j}^{(k+1)} = \mathbf{z}_{i|j}^{(k)} - \rho \left(\mathbf{A}_{i|j}\mathbf{x}_i^{(k+1)} - \frac{\mathbf{b}_{ij}}{2} \right)$
    $\mathbf{y}_{i|j}^{(k+1)} = 2\boldsymbol{\lambda}_{i|j}^{(k+1)} - \mathbf{z}_{i|j}^{(k)}$
  end for
end for
for all $i \in V, j \in \mathcal{N}(i)$ **do** ▷ Transmit Variables
  $\mathbf{Node}_j \leftarrow \mathbf{Node}_i \left(\mathbf{y}_{i|j}^{(k+1)} \right)$
end for
for all $i \in V, j \in \mathcal{N}(i)$ **do** ▷ Auxiliary Update
  $\mathbf{z}_{i|j}^{(k+1)} = \mathbf{y}_{j|i}^{(k+1)}$
end for
end for

4.3.1. FROM A PROTOTYPE OPTIMIZATION PROBLEM TO EQUIVALENT DUAL FORM

To derive the proposed modified algorithm we begin by mimicking the derivation of PDMM provided in Chapter 3. Firstly, consider a simple, undirected network consisting of N nodes. The associated graphical model of this network is given by $G(V,E)$ where $V = \{1,...,N\}$ and E denote the set of nodes and undirected edges respectively. Assume that each node is equipped with a function $f_i \in \Gamma_0\left(\mathbb{R}^{M_i}\right)\ \forall i \in V$ parameterized by a local variable $\mathbf{x}_i \in \mathbb{R}^{M_i}$. Here Γ_0 is used to denote the set of closed, convex and proper functions. Under this model, consider solving the following optimization problem in a distributed manner.

$$\min_{\mathbf{x}_i\, \forall\, i \in V} \sum_{i \in V} f_i(\mathbf{x}_i) \quad \text{s.t}\, \mathbf{A}_{i|j}\mathbf{x}_i + \mathbf{A}_{j|i}\mathbf{x}_j = \mathbf{b}_{ij}\ \forall\ (i,j) \in E. \tag{4.1}$$

Here $\mathbf{A}_{i|j} \in \mathbb{R}^{M_{i,j} \times M_i}$ and $\mathbf{b}_{ij} \in \mathbb{R}^{M_{i,j}}$. Furthermore, let $M_V = \sum_{i \in V} M_i$ and $M_E = \sum_{(i,j) \in E} M_{i,j}$. Note the distinction between the subscripts $i|j$ and ij. The prior is a *directional* identifier used to denote the directed edge from node i to node j, while the later is an *undirected* identifier.

The associated Lagrange dual [48] of (4.1) is given by

$$\min_{\boldsymbol{\nu}} \quad \sum_{i \in V} \left(f_i^* \left(\sum_{j \in \mathcal{N}(i)} \mathbf{A}_{i|j}^T \boldsymbol{\nu}_{ij} \right) - \sum_{j \in \mathcal{N}(i)} \frac{\mathbf{b}_{ij}}{2}^T \boldsymbol{\nu}_{ij} \right), \tag{4.2}$$

where each $\boldsymbol{\nu}_{ij} \in \mathbb{R}^{M_{i,j}}$ denotes the dual vector variable associated with the constraint function along edge (i,j), $\mathcal{N}(i) = \{j \in V \mid (i,j) \in E\}$ is the set of neighbors of node i

and f_i^* is the Fenchel conjugate of f_i. By inspection, each $\boldsymbol{v}_{ij}$ in (4.2) is utilized in two conjugate functions, f_i^* and f_j^* resulting in a coupling between neighboring nodes.

To address the linking of the objective terms, the dimension of the dual problem can be lifted by introducing additional directed edge variables $\boldsymbol{\lambda}_{i|j}$ and $\boldsymbol{\lambda}_{j|i}$. These are then constrained so that at optimality $\boldsymbol{\lambda}_{i|j} = \boldsymbol{\lambda}_{j|i}$, leading to what we term as the *extended dual* of Eq (4.1)

$$\begin{aligned} \min_{\boldsymbol{\lambda}} \quad & \sum_{i\in V}\left(f_i^*\left(\sum_{j\in\mathcal{N}(i)}\mathbf{A}_{i|j}^T\boldsymbol{\lambda}_{i|j}\right)-\sum_{j\in\mathcal{N}(i)}\frac{\mathbf{b}_{ij}}{2}^T\boldsymbol{\lambda}_{i|j}\right) \\ \text{s.t.} \quad & \boldsymbol{\lambda}_{i|j}=\boldsymbol{\lambda}_{j|i} \quad \forall i\in V, j\in\mathcal{N}(i). \end{aligned} \tag{4.3}$$

We denote by $\boldsymbol{\lambda}$ the stacked vector of all $\boldsymbol{\lambda}_{i|j}$. The ordering of this stacking is given by $1|2<1|3<\cdots<1|N<2|1<2|3<\cdots<N|N-1$. In the case of a fully connected network the vector $\boldsymbol{\lambda}\in\mathbb{R}^{M_E}$ is given by

$$\boldsymbol{\lambda}=\left[\boldsymbol{\lambda}_{1|2};\cdots;\boldsymbol{\lambda}_{1|N};\boldsymbol{\lambda}_{2|1};\cdots,\boldsymbol{\lambda}_{N|N-1}\right].$$

For the primal constraints we introduce the matrix $\mathbf{C}\in\mathbb{R}^{M_E\times M_V}$ and vector $\mathbf{d}\in\mathbb{R}^{M_E}$ where in the case of a fully connected network

$$\begin{aligned} \mathbf{C}&=\begin{bmatrix}\mathbf{C}_1 & \cdots & \mathbf{0}\\ \vdots & \ddots & \vdots\\ \mathbf{0} & \cdots & \mathbf{C}_N\end{bmatrix},\ \mathbf{d}=\left[\mathbf{d}_1;\mathbf{d}_2;\cdots;\mathbf{d}_N\right]\\ \mathbf{C}_i&=\left[\mathbf{A}_{i|1};\mathbf{A}_{i|2};\cdots;\mathbf{A}_{i|i-1};\mathbf{A}_{i|i+1};\cdots;\mathbf{A}_{i|N}\right]\forall i\in V\\ \mathbf{d}_i&=\frac{1}{2}\left[\mathbf{b}_{i1};\mathbf{b}_{i2};\cdots;\mathbf{b}_{i(i-1)};\mathbf{b}_{i(i+1)};\mathbf{b}_{i2};\cdots;\mathbf{b}_{iN}\right]\forall i\in V. \end{aligned}$$

For other network topologies the unnecessary rows of $\boldsymbol{\lambda}$, $\mathbf{C}$ and $\mathbf{d}$, corresponding to non-existent edges, can be removed.

We further introduce the symmetric permutation matrix $\mathbf{P}$ that maps between each pair of variables $\boldsymbol{\lambda}_{i|j}$ and $\boldsymbol{\lambda}_{j|i}$ so that the constraints in (4.3) can be rewritten as $\boldsymbol{\lambda}=\mathbf{P}\boldsymbol{\lambda}$. Finally, we define the function

$$f:\mathbb{R}^{M_V}\mapsto\mathbb{R},\ \mathbf{x}\mapsto\sum_{i\in V}f_i(\mathbf{x}_i),$$

as the sum of all local functions where $\mathbb{R}^{M_V}=\mathbb{R}^{M_1}\times\mathbb{R}^{M_2}\times\ldots\times\mathbb{R}^{M_N}$. It follows that (4.3) is equivalent to

$$\min_{\boldsymbol{\lambda}} \quad f^*(\mathbf{C}^T\boldsymbol{\lambda})-\mathbf{d}^T\boldsymbol{\lambda}+\iota_{\ker(\mathbf{I}-\mathbf{P})}(\boldsymbol{\lambda}), \tag{4.4}$$

where $\iota_{\ker(\mathbf{I}-\mathbf{P})}$ is the indicator function of the edge based constraints defined as

$$\iota_{\ker(\mathbf{I}-\mathbf{P})}(\mathbf{y})=\begin{cases}0 & (\mathbf{I}-\mathbf{P})\mathbf{y}=0\\ +\infty & \text{otherwise.}\end{cases}$$

4.3.2. FROM AN UNCONSTRAINED OPTIMIZATION PROBLEM TO A NONEXPANSIVE OPERATOR

From the unconstrained optimization problem in (4.4) we now introduce a method of solving (4.1) in a distributed manner. Unlike the standard PDMM algorithm we shall do so via a primal-dual interpretation which ultimately can be shown to naturally introduce the primal regularization alluded to at the onset of this section.

The first step in forming the aforementioned primal-dual reformulation is to recast (4.4) as an equivalent monotonic inclusion. To achieve this, we can note that as the operator $\mathbf{I}-\mathbf{P}$ is continuous, $\ker(\mathbf{I}-\mathbf{P})$ is a closed subspace. It follows from [34] (Example 1.25) that $\iota_{\ker(\mathbf{I}-\mathbf{P})} \in \Gamma_0$. By combining this point with Theorem 13.37 and Proposition 13.3, both of [34], and the assumption that $f \in \Gamma_0$, it follows that (4.4) is an unconstrained optimization problem of a closed, proper and convex function. By Fermat's Rule (Theorem 16.3) [34], it follows that a point $\boldsymbol{\lambda}^*$ is a minimizer of (4.4) if and only if

$$\mathbf{0} \in \mathbf{C}\partial f^*\left(\mathbf{C}^T\boldsymbol{\lambda}\right) - \mathbf{d} + \partial \iota_{\ker(\mathbf{I}-\mathbf{P})}(\boldsymbol{\lambda}), \tag{4.5}$$

where $\partial \iota_{\ker(\mathbf{I}-\mathbf{P})}$ is a normal cone operator [47].

Equivalently this can be rephrased in a primal-dual monotonic inclusion form by noting the equivalence of (4.5) and

$$\mathbf{0} \in \mathbf{C}\mathbf{x} - \mathbf{d} + \partial \iota_{\ker(\mathbf{I}-\mathbf{P})}(\boldsymbol{\lambda}),\ \mathbf{x} \in \partial f^*\left(\mathbf{C}^T\boldsymbol{\lambda}\right).$$

Due to the relationship between the subdifferential of conjugates of CCP functions and their inverses, by introducing the scalar β, it can be shown that

$$\mathbf{0} \in \mathbf{C}\mathbf{x} - \mathbf{d} + \partial \iota_{\ker(\mathbf{I}-\mathbf{P})}(\boldsymbol{\lambda}),\ \mathbf{0} \in \beta\left(\partial f(\mathbf{x}) - \mathbf{C}^T\boldsymbol{\lambda}\right),$$

where we restrict $\beta \in (0, +\infty)$. With some manipulation, this set of monotonic inclusions can be rewritten in the concise form

$$\mathbf{0} \in \begin{bmatrix} \mathbf{0} & \mathbf{C} \\ -\beta\mathbf{C}^T & \beta\partial f \end{bmatrix}\begin{bmatrix}\boldsymbol{\lambda}\\ \mathbf{x}\end{bmatrix} - \begin{bmatrix}\mathbf{d}\\ \mathbf{0}\end{bmatrix} + \begin{bmatrix} \partial\delta_{(\mathbf{I}-\mathbf{P})} & \mathbf{0} \\ \mathbf{0} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{\lambda}\\ \mathbf{x}\end{bmatrix}. \tag{4.6}$$

The modified algorithm proposed in this chapter aims to solve (4.6) via operator splitting methods. For compactness in the following sections, we define the two operators

$$\mathbf{T}_1\left(\begin{bmatrix}\boldsymbol{\lambda}\\ \mathbf{x}\end{bmatrix}\right) = \begin{bmatrix} \mathbf{0} & \mathbf{C} \\ -\beta\mathbf{C}^T & \beta\partial f \end{bmatrix}\begin{bmatrix}\boldsymbol{\lambda}\\ \mathbf{x}\end{bmatrix} - \begin{bmatrix}\mathbf{d}\\ \mathbf{0}\end{bmatrix},\ \mathbf{T}_2\left(\begin{bmatrix}\boldsymbol{\lambda}\\ \mathbf{x}\end{bmatrix}\right) = \begin{bmatrix} \partial\delta_{(\mathbf{I}-\mathbf{P})} & \mathbf{0} \\ \mathbf{0} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{\lambda}\\ \mathbf{x}\end{bmatrix}.$$

Before moving to the algorithm itself, firstly we will demonstrate a space for which the aforementioned operators are monotone. For this purpose, we define the metric

$$\mathscr{J} = \begin{bmatrix}\beta\mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{bmatrix}$$

which defines an inner product space with inner product $\langle \mathbf{x}, \mathbf{y}\rangle_{\mathscr{J}} = \mathbf{x}^T\mathscr{J}\mathbf{y}$ and induced norm $\|\mathbf{y}\|_{\mathscr{J}} = \sqrt{\mathbf{y}^T\mathscr{J}\mathbf{y}}$. It follows that, given two vectors $\mathbf{z}_1, \mathbf{z}_2$,

$$\left\langle \mathbf{T}_1\left(\begin{bmatrix}\boldsymbol{\lambda}_1\\ \mathbf{x}_1\end{bmatrix}\right) - \mathbf{T}_1\left(\begin{bmatrix}\boldsymbol{\lambda}_2\\ \mathbf{x}_2\end{bmatrix}\right), \begin{bmatrix}\boldsymbol{\lambda}_1\\ \mathbf{x}_1\end{bmatrix} - \begin{bmatrix}\boldsymbol{\lambda}_2\\ \mathbf{x}_2\end{bmatrix}\right\rangle_{\mathscr{J}} = \beta\langle \partial f(\mathbf{x}_1) - \partial f(\mathbf{x}_2), \mathbf{x}_1 - \mathbf{x}_2\rangle \geq 0,$$

where the inequality follows from the monotonicity of ∂f such that $\mathbf{T}_1$ is monotone in the proposed space. Futhermore, given the same two vectors, it follows that

$$\left\langle \mathbf{T}_2\left(\begin{bmatrix}\boldsymbol{\lambda}_1\\ \mathbf{x}_1\end{bmatrix}\right)-\mathbf{T}_2\left(\begin{bmatrix}\boldsymbol{\lambda}_2\\ \mathbf{x}_2\end{bmatrix}\right),\begin{bmatrix}\boldsymbol{\lambda}_1\\ \mathbf{x}_1\end{bmatrix}-\begin{bmatrix}\boldsymbol{\lambda}_2\\ \mathbf{x}_2\end{bmatrix}\right\rangle_{\mathscr{I}}=\beta\left\langle \partial\delta_{(\mathbf{I}-\mathbf{P})}(\boldsymbol{\lambda}_1)-\partial\delta_{(\mathbf{I}-\mathbf{P})}(\boldsymbol{\lambda}_2),\boldsymbol{\lambda}_1-\boldsymbol{\lambda}_2\right\rangle\geq 0,$$

such that $\mathbf{T}_2$ is also monotone. The maximality of both operators follows from the maximality of the original operators.

As the summation of two maximal monotone operators, it follows that (4.6) can be solved via a number of classic monotone splitting methods. Whilst not strictly guaranteed to converge for general maximal monotone operators, in an effort to mimic the derivation of PDMM, in this case we will consider applying Peaceman-Rachford splitting to (4.6). Using the results of Section 2.4.2, it follows that (4.6) can be equivalently expressed by the fixed point inclusion

$$\begin{bmatrix}\mathbf{z}_{\boldsymbol{\lambda}}\\ \mathbf{z}_{\mathbf{x}}\end{bmatrix}=\mathbf{R}_{\rho,\mathbf{T}_2}\circ\mathbf{R}_{\rho,\mathbf{T}_1}\left(\begin{bmatrix}\mathbf{z}_{\boldsymbol{\lambda}}\\ \mathbf{z}_{\mathbf{x}}\end{bmatrix}\right),\ \begin{bmatrix}\mathbf{z}_{\boldsymbol{\lambda}}\\ \mathbf{z}_{\mathbf{x}}\end{bmatrix}=\mathbf{J}_{\rho,\mathbf{T}_2}\circ\mathbf{R}_{\rho,\mathbf{T}_1}\left(\begin{bmatrix}\boldsymbol{\lambda}\\ \mathbf{x}\end{bmatrix}\right),\tag{4.7}$$

where the operators $\mathbf{R}_{\rho,\mathbf{T}_1}$ and $\mathbf{R}_{\rho,\mathbf{T}_2}$ are non-expansive with respect to the inner product space induced by the metric $\mathscr{I}$. The introduced vectors $\mathbf{z}_{\boldsymbol{\lambda}}$ and $\mathbf{z}_{\mathbf{x}}$ are *auxiliary* variables associated with $\boldsymbol{\lambda}$ and $\mathbf{x}$ respectively.

As demonstrated in Chapter 2, PR splitting is in general not sufficient for guaranteed convergence. Whilst operator averaging, such as that used in Douglas-Rachford splitting is commonly used to address this, it can also reduce the convergence rate of an algorithm, essentially acting as a sort of compromise between new and old iterates. In this work we demonstrate a hybrid type approach which aims to address this point by providing guaranteed convergence whilst only introducing an averaging over the primal variables of (4.7). From a distributed perspective, the motivation for this approach is that like the original PDMM algorithm we can show that any edge based operations are purely exchanges of information within the network and thus do not require any direct collaboration between nodes. The resulting iterative approach is given by

$$\begin{bmatrix}\mathbf{z}_{\boldsymbol{\lambda}}^{(k+1)}\\ \mathbf{z}_{\mathbf{x}}^{(k+1)}\end{bmatrix}=\left(\begin{bmatrix}\mathbf{0}&\mathbf{0}\\ \mathbf{0}&\frac{1}{2}\mathbf{I}\end{bmatrix}+\begin{bmatrix}\mathbf{I}&\mathbf{0}\\ \mathbf{0}&\frac{1}{2}\mathbf{I}\end{bmatrix}\mathbf{R}_{\rho,\mathbf{T}_2}\circ\mathbf{R}_{\rho,\mathbf{T}_1}\right)\left(\begin{bmatrix}\mathbf{z}_{\boldsymbol{\lambda}}^{(k)}\\ \mathbf{z}_{\mathbf{x}}^{(k)}\end{bmatrix}\right)$$

4.3.3. SIMPLIFYING THE COMPUTATION OF REFLECTED RESOLVENTS

To allow for the computation of the iterative algorithm defined above we introduce the two following Lemmas which facilitate the computation of the reflected resolvents.

Lemma 4.3.1.

$$\mathbf{R}_{\rho,\mathbf{T}_1}\left(\begin{bmatrix}\mathbf{z}_{\boldsymbol{\lambda}}^{(k)}\\ \mathbf{z}_{\mathbf{x}}^{(k)}\end{bmatrix}\right)=\begin{bmatrix}\mathbf{z}_{\boldsymbol{\lambda}}^{(k)}\\ \mathbf{z}_{\mathbf{x}}^{(k)}\end{bmatrix}=2\begin{bmatrix}\boldsymbol{\lambda}^{(k)+1}\\ \mathbf{x}^{(k+1)}\end{bmatrix}-\begin{bmatrix}\mathbf{z}_{\boldsymbol{\lambda}}^{(k)}\\ \mathbf{z}_{\mathbf{x}}^{(k)}\end{bmatrix},$$

where

$$\begin{aligned}\mathbf{x}^{(k+1)}=&\arg\min_{\mathbf{x}}\left(f(\mathbf{x})-\left\langle\mathbf{C}^T\mathbf{z}_{\boldsymbol{\lambda}}^{(k)},\mathbf{x}\right\rangle+\frac{\rho}{2}\|\mathbf{C}\mathbf{x}-\mathbf{d}\|^2+\frac{1}{2\rho\beta}\left\|\mathbf{x}-\mathbf{z}_{\mathbf{x}}^{(k)}\right\|^2\right)\\ \boldsymbol{\lambda}^{(k+1)}=&\mathbf{z}_{\boldsymbol{\lambda}}^{(k)}-\rho\left(\mathbf{C}\mathbf{x}^{(k+1)}-\mathbf{d}\right),\ \mathbf{y}_{\boldsymbol{\lambda}}^{(k+1)}=2\boldsymbol{\lambda}^{(k+1)}-\mathbf{z}^{(k)},\ \mathbf{y}_{\mathbf{x}}^{(k+1)}=2\mathbf{x}^{(k+1)}-\mathbf{z}_{\mathbf{x}}^{(k)}\end{aligned}$$

The proof of this Lemma can be found in Appendix 4.A

Lemma 4.3.2.

$$\mathbf{R}_{\rho,\mathbf{T}_2}\left(\begin{bmatrix}\mathbf{y}_{\boldsymbol{\lambda}}^{(k+1)}\\ \mathbf{y}_{\mathbf{x}}^{(k+1)}\end{bmatrix}\right)=\begin{bmatrix}\mathbf{z}_{\boldsymbol{\lambda}}^{(k+1)}\\ \mathbf{z}_{\mathbf{x}}^{(k+1)}\end{bmatrix}=\begin{bmatrix}\mathbf{P} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{y}_{\boldsymbol{\lambda}}^{(k+1)}\\ \mathbf{y}_{\mathbf{x}}^{(k+1)}\end{bmatrix},$$

The proof for this Lemma can be found in Appendix 4.B

4.3.4. THE MODIFIED PDMM ALGORITHM (M-PDMM)

By applying Lemmas 4.3.1 and 4.3.2 and defining the scalar variable $\gamma=\frac{1}{\rho\beta}$, the modified PDMM scheme is summarized in Algorithm 4.

Algorithm 4 Modified PDMM (m-PDMM)

Initialise: $\mathbf{z}^{(0)}\in\mathbb{R}^{M_E}$
for k=0,..., **do**
  $\mathbf{x}^{(k+1)}=\underset{\mathbf{x}}{\operatorname{argmin}}\left(f(\mathbf{x})-\left\langle\mathbf{C}^T\mathbf{z}_{\boldsymbol{\lambda}}^{(k)},\mathbf{x}\right\rangle+\frac{\rho}{2}\|\mathbf{C}\mathbf{x}-\mathbf{d}\|^2+\frac{\gamma}{2}\left\|\mathbf{x}-\mathbf{z}_{\mathbf{x}}^{(k)}\right\|^2\right)$
  $\boldsymbol{\lambda}^{(k+1)}=\mathbf{z}_{\boldsymbol{\lambda}}^{(k)}-\rho\left(\mathbf{C}\mathbf{x}^{(k+1)}-\mathbf{d}\right)$
  $\mathbf{y}_{\boldsymbol{\lambda}}^{(k+1)}=2\boldsymbol{\lambda}^{(k+1)}-\mathbf{z}^{(k)}$
  $\mathbf{y}_{\mathbf{x}}^{(k+1)}=2\mathbf{x}^{(k+1)}-\mathbf{z}_{\mathbf{x}}^{(k)}$
  $\mathbf{z}_{\boldsymbol{\lambda}}^{(k+1)}=\mathbf{P}\mathbf{y}_{\boldsymbol{\lambda}}^{(k+1)}$
  $\mathbf{z}_{\mathbf{x}}^{(k+1)}=\frac{1}{2}\left(\mathbf{y}_{\mathbf{x}}^{(k+1)}+\mathbf{z}_{\mathbf{x}}^{(k)}\right)=\mathbf{x}^{(k+1)}$.
end for

Aside from the introduction of the additional auxiliary $\mathbf{z}_x$ variables, the only distinction between the standard PDMM (Algorithm 3) and the proposed modification (Algorithm 4) is the inclusion of the additional primal regularization alluded to at the beginning of this chapter. The proposed approach also provides flexibility in how this regularization is utilized through the inclusion of the additional parameter γ. As with the stepsize parameter ρ, the choice of γ has a dramatic effect on the convergence of the algorithm as it directly controls the influence that the primal regularization plays at each iteration. Importantly, so long as $\gamma>0$ we demonstrate in the coming sections, that this modification is sufficient to guarantee that the algorithm converges to a primal optimal point.

4.4. ON THE GUARANTEED CONVERGENCE OF THE M-PDMM ALGORITHM

We now move to demonstrating the convergence characteristics of the modified PDMM algorithm (m-PDMM) for the general case of CCP functions. Notably, this proof is broken down into four key stages which are ultimately combined to ensure that a primal optimal point is obtained. These stages are outlined in the coming subsections.

4.4.1. CONVERGENCE OF THE PRIMAL VARIABLES TO A LIMIT STATE

The first step in the convergence proof is to verify that the primal variables $\mathbf{x}$ converge to a fixed vector. To demonstrate this fact, we begin by defining the matrix

$$\tilde{\mathscr{I}} = \begin{bmatrix} \beta\mathbf{I} & \mathbf{0} \\ \mathbf{0} & 2\mathbf{I} \end{bmatrix}.$$

We will use this matrix as a metric for a new quadratic form $\|\mathbf{x}\|^2_{\tilde{\mathscr{I}}} = \mathbf{x}^T\tilde{\mathscr{I}}\mathbf{x}$. Furthermore we will assume that there exists a $\mathbf{z}^* \in \text{fix}\left(\mathbf{R}_{\rho,\mathbf{T}_2} \circ \mathbf{R}_{\rho,\mathbf{T}_1}\right)$ i.e. that the auxiliary fixed point set is non-empty. Here, $\mathbf{z}^* = \left[\mathbf{z}_{\boldsymbol{\lambda}}^{*T}, \mathbf{z}_{\mathbf{x}}^{*T}\right]^T$ is used to denotes the stacked vector of auxiliary variables. Similarly, from here on out $\mathbf{z} = \left[\mathbf{z}_{\boldsymbol{\lambda}}^T, \mathbf{z}_{\mathbf{x}}^T\right]^T$ will be used to denote other such stacked vectors. Using these definitions and by applying the m-PDMM algorithm it follows that for any given iteration k, that

$$\begin{aligned}\left\|\mathbf{z}^{(k+1)} - \mathbf{z}^*\right\|^2_{\tilde{\mathscr{I}}} &= \beta\left\|\mathbf{z}_{\boldsymbol{\lambda}}^{(k+1)} - \mathbf{z}_{\boldsymbol{\lambda}}^*\right\|^2 + 2\left\|\mathbf{z}_{\mathbf{x}}^{(k+1)} - \mathbf{z}_{\mathbf{x}}^*\right\|^2 \\ &\leq \beta\left\|\mathbf{z}_{\boldsymbol{\lambda}}^{(k+1)} - \mathbf{z}_{\boldsymbol{\lambda}}^*\right\|^2 + \left\|\mathbf{y}_{\mathbf{x}}^{(k+1)} - \mathbf{z}_{\mathbf{x}}^*\right\|^2 + \left\|\mathbf{z}_{\mathbf{x}}^{(k)} - \mathbf{z}_{\mathbf{x}}^*\right\|^2 - \frac{1}{2}\left\|\mathbf{y}_{\mathbf{x}}^{(k+1)} - \mathbf{z}_{\mathbf{x}}^{(k)}\right\|^2\end{aligned}$$

where in the first line we have used the definition of $\tilde{\mathscr{I}}$, in the second line we have used the averaged nature of the $\mathbf{z}_{\mathbf{x}}$ variables and the inequality in (2.2). As $\mathbf{y}_{\mathbf{x}}^{(k+1)} = \mathbf{z}_{\mathbf{x}}^{(k+1)}$ from the definition of the primal variables, it therefore follows that

$$\begin{aligned}\left\|\mathbf{z}^{(k+1)} - \mathbf{z}^*\right\|^2_{\tilde{\mathscr{I}}} &\leq \left\|\mathbf{z}^{(k+1)} - \mathbf{z}^*\right\|^2_{\mathscr{I}} + \left\|\mathbf{z}_{\mathbf{x}}^{(k)} - \mathbf{z}_{\mathbf{x}}^*\right\|^2 - \frac{1}{2}\left\|\mathbf{y}_{\mathbf{x}}^{(k+1)} - \mathbf{z}_{\mathbf{x}}^{(k)}\right\|^2 \\ &\leq \left\|\mathbf{z}^{(k)} - \mathbf{z}^*\right\|^2_{\mathscr{I}} + \left\|\mathbf{z}_{\mathbf{x}}^{(k)} - \mathbf{z}_{\mathbf{x}}^*\right\|^2 - \frac{1}{2}\left\|\mathbf{y}_{\mathbf{x}}^{(k+1)} - \mathbf{z}_{\mathbf{x}}^{(k)}\right\|^2 \\ &\leq \left\|\mathbf{z}^{(k)} - \mathbf{z}^*\right\|^2_{\tilde{\mathscr{I}}} - \frac{1}{2}\left\|\mathbf{y}_{\mathbf{x}}^{(k+1)} - \mathbf{z}_{\mathbf{x}}^{(k)}\right\|^2 \\ &= \left\|\mathbf{z}^{(k)} - \mathbf{z}^*\right\|^2_{\tilde{\mathscr{I}}} - 2\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right\|^2,\end{aligned}$$

where in the second line we have used the nonexpansiveness of the m-PDMM updates in the metric space defined by $\mathscr{I}$, in the third line we have used the definition of $\tilde{\mathscr{I}}$ and the final inequality stems from the primal $\mathbf{x}$ updates in Algorithm 4.

It follows from [34, Definition 5.1] that the auxiliary updates $(\mathbf{z}^{(k+1)})$ are Fejér Monotone with respect to $\mathbf{z}^*$ and furthermore that the sequence $\left(\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right\|^2\right)_{k\in N}$ is finitely summable and converges. We therefore find that for any sequence of primal variable $\left(\mathbf{x}^{(k)}\right)_{k\in\mathbb{N}}$ generated by the m-PDMM algorithm that $\exists\tilde{\mathbf{x}} \mid \mathbf{x}^{(k)} \to \tilde{\mathbf{x}}$. In other words, regardless of initialization, the primal variables of the m-PDMM algorithm will always converge to a limit state.

4.4.2. FEASIBILITY OF THE PRIMAL LIMIT STATE

Given the convergence of the primal variables to a limit state, we now move to proving that such a point is primal feasible. To do so, consider any fixed point $\mathbf{z}^*$ of the m-PDMM algorithm and an iteration number k. From the definitions of the primal and dual update

steps in 4.3.1, it follows that

$$\mathbf{z}_{\boldsymbol{\lambda}}^{(k+1)} = \mathbf{P}\left(\mathbf{z}_{\boldsymbol{\lambda}}^{(k)} - 2\rho\left(\mathbf{C}\mathbf{x}^{(k+1)} - \mathbf{d}\right)\right),$$

and thus that for any two successive iterations

$$\mathbf{z}_{\boldsymbol{\lambda}}^{(k+2)} = \mathbf{z}_{\boldsymbol{\lambda}}^{(k)} - 2\rho\left(\mathbf{P}\mathbf{C}\mathbf{x}^{(k+2)} + \mathbf{C}\mathbf{x}^{(k+1)} - 2\mathbf{d}\right).$$

For any $M \in \mathbb{N}$ it follows that the corresponding $2M$-step updates are given by

$$\mathbf{z}_{\boldsymbol{\lambda}}^{(k+2)} = \mathbf{z}_{\boldsymbol{\lambda}}^{(k)} - 2\rho\sum_{i=k+2}^{k+2M}\left(\mathbf{P}\mathbf{C}\mathbf{x}^{(i)} + \mathbf{C}\mathbf{x}^{(i-1)} - 2\mathbf{d}\right). \tag{4.8}$$

With this equality in mind, consider the auxiliary squared norm term given by

$$\begin{aligned}\left\|\mathbf{z}^{(k+2M)} - \mathbf{z}^*\right\|_{\tilde{\mathscr{F}}}^2 &= \beta\left\|\mathbf{z}_{\boldsymbol{\lambda}}^{(k+2)} - \mathbf{z}_{\boldsymbol{\lambda}}^*\right\|^2 + 2\left\|\mathbf{z}_{\mathbf{x}}^{(k+2M)} - \mathbf{z}_{\mathbf{x}}^*\right\|^2 \\ &= \beta\left\|\mathbf{z}_{\boldsymbol{\lambda}}^{(k)} - 2\rho\sum_{i=k+2}^{k+2M}\left(\mathbf{P}\mathbf{C}\mathbf{x}^{(i)} + \mathbf{C}\mathbf{x}^{(i-1)} - 2\mathbf{d}\right) - \mathbf{z}_{\boldsymbol{\lambda}}^*\right\|^2 + 2\left\|\mathbf{z}_{\mathbf{x}}^{(k+2)} - \mathbf{z}_{\mathbf{x}}^*\right\|^2\end{aligned}$$

Combining this with (4.8), it follows that

$$\begin{aligned}\left\|\mathbf{z}^{(k+2M)} - \mathbf{z}^*\right\|_{\tilde{\mathscr{F}}}^2 &= \beta\left\|\mathbf{z}_{\boldsymbol{\lambda}}^{(k)} - \mathbf{z}_{\boldsymbol{\lambda}}^*\right\|^2 - 2\left\langle 2\rho\sum_{i=k+2}^{k+2M}\left(\mathbf{P}\mathbf{C}\mathbf{x}^{(i)} + \mathbf{C}\mathbf{x}^{(i-1)} - 2\mathbf{d}\right), \mathbf{z}_{\boldsymbol{\lambda}}^{(k)} - \mathbf{z}_{\boldsymbol{\lambda}}^*\right\rangle \\ &\quad + \left\|2\rho\sum_{i=k+2}^{k+2M}\left(\mathbf{P}\mathbf{C}\mathbf{x}^{(i)} + \mathbf{C}\mathbf{x}^{(i-1)} - 2\mathbf{d}\right)\right\|^2 + 2\left\|\mathbf{z}_{\mathbf{x}}^{(k+2)} - \mathbf{z}_{\mathbf{x}}^*\right\|^2 \\ &= \left\|\mathbf{z}^{(k)} - \mathbf{z}^*\right\|_{\tilde{\mathscr{F}}}^2 - 2\left\langle 2\rho\sum_{i=k+2}^{k+2M}\left(\mathbf{P}\mathbf{C}\mathbf{x}^{(i)} + \mathbf{C}\mathbf{x}^{(i-1)} - 2\mathbf{d}\right), \mathbf{z}_{\boldsymbol{\lambda}}^{(k)} - \mathbf{z}_{\boldsymbol{\lambda}}^*\right\rangle \\ &\quad + \left\|2\rho\sum_{i=k+2}^{k+2M}\left(\mathbf{P}\mathbf{C}\mathbf{x}^{(i)} + \mathbf{C}\mathbf{x}^{(i-1)} - 2\mathbf{d}\right)\right\|^2 + 2\left\|\mathbf{z}_{\mathbf{x}}^{(k+2)} - \mathbf{z}_{\mathbf{x}}^*\right\|^2 - 2\left\|\mathbf{z}_{\mathbf{x}}^{(k)} - \mathbf{z}_{\mathbf{x}}^*\right\|^2,\end{aligned}$$

where in the final line we have again used the definition of the auxiliary $\mathbf{z}$ variables of m-PDMM. Moving all norm terms to the left hand side and taking limits it follows that

$$\begin{aligned}&\lim_{k\to\infty}\left(\left\|\mathbf{z}^{(k+2M)} - \mathbf{z}^*\right\|_{\tilde{\mathscr{F}}}^2 - \left\|\mathbf{z}^{(k)} - \mathbf{z}^*\right\|_{\tilde{\mathscr{F}}}^2 - 2\left(\left\|\mathbf{z}_{\mathbf{x}}^{(k+2)} - \mathbf{z}_{\mathbf{x}}^*\right\|^2 - \left\|\mathbf{z}_{\mathbf{x}}^{(k)} - \mathbf{z}_{\mathbf{x}}^*\right\|^2\right)\right) \\ &= \lim_{k\to\infty}\left(-2\left\langle 2\rho\sum_{i=k+2}^{k+2M}\left(\mathbf{P}\mathbf{C}\mathbf{x}^{(i)} + \mathbf{C}\mathbf{x}^{(i-1)} - 2\mathbf{d}\right), \mathbf{z}_{\boldsymbol{\lambda}}^{(k)} - \mathbf{z}_{\boldsymbol{\lambda}}^*\right\rangle + \left\|2\rho\sum_{i=k+2}^{k+2M}\left(\mathbf{P}\mathbf{C}\mathbf{x}^{(i)} + \mathbf{C}\mathbf{x}^{(i-1)} - 2\mathbf{d}\right)\right\|^2\right) \\ &= -2\left\langle 2\rho M\left((\mathbf{P}\mathbf{C} + \mathbf{C})\,\tilde{\mathbf{x}} - 2\mathbf{d}\right), \lim_{k\to\infty}\left(\mathbf{z}_{\boldsymbol{\lambda}}^{(k)} - \mathbf{z}_{\boldsymbol{\lambda}}^*\right)\right\rangle + \left\|2\rho M\left((\mathbf{P}\mathbf{C} + \mathbf{C})\,\tilde{\mathbf{x}} - 2\mathbf{d}\right)\right\|^2 = 0\end{aligned}$$

where we have used the fact that $\left(\left\|\mathbf{z}^{(k)}-\mathbf{z}^*\right\|_{\tilde{\mathscr{F}}}^2\right)_{k\in\mathbb{N}}$ is a convergent sequence and that $\mathbf{x}^{(k)}\to\tilde{\mathbf{x}}$. By simple elimination of scalar coefficients, it follows that

$$\left\langle (\mathbf{PC}+\mathbf{C})\,\tilde{\mathbf{x}}-2\mathbf{d},\ \lim_{k\to\infty}\left(\mathbf{z}_{\boldsymbol{\lambda}}^{(k)}-\mathbf{z}_{\boldsymbol{\lambda}}^*\right)\right\rangle = \rho M\,\|(\mathbf{PC}+\mathbf{C})\,\tilde{\mathbf{x}}-2\mathbf{d}\|^2$$

which can only hold simultaneously for all $M\in N_+$ if $(\mathbf{PC}+\mathbf{C})\,\tilde{\mathbf{x}}-2\mathbf{d}=\mathbf{0}$, i.e. that $\tilde{\mathbf{x}}$ must be a primal feasible vector. Thus the primal variables of PDMM are guaranteed to converge to a limit state which is a feasible solution of (4.1).

4.4.3. ON THE LIMIT STATES OF THE DUAL VARIABLES

Given the convergence of the primal variables to a primal feasible limit state, the next step in the convergence proof is to demonstrate the convergence of the $\boldsymbol{\lambda}$ variables to their limit states. Unlike the primal variables, in this section we show that the dual limit state need not converge to a limit state but that the odd and even iterations must. Fortunately, it turns out that this property is still sufficient to guarantee optimality of the primal limit states.

To demonstrate convergence of the even and odd $\boldsymbol{\lambda}$ iterates to their respective limit states, we begin by noting that from the two step update of the auxiliary variables, that

$$\mathbf{z}_{\boldsymbol{\lambda}}^{(k+2)} = \mathbf{z}_{\boldsymbol{\lambda}}^{(k)} - 2\rho\left(\mathbf{PCx}^{(k+2)}+\mathbf{Cx}^{(k+1)}-2\mathbf{d}\right) \tag{4.9}$$

Using the feasibility of the primal variables in the limit, it follows that,

$$\lim_{k\to\infty}\left(\mathbf{z}_{\boldsymbol{\lambda}}^{(k+2)}-\mathbf{z}_{\boldsymbol{\lambda}}^{(k)}\right) = \lim_{k\to\infty}\left(-2\rho\left(\mathbf{PCx}^{(k+2)}+\mathbf{Cx}^{(k+1)}-2\mathbf{d}\right)\right) = -2\rho\left((\mathbf{PC}+\mathbf{C})\,\tilde{\mathbf{x}}-2\mathbf{d}\right)=\mathbf{0}.$$

Assuming, without loss of generality, that k is even, we therefore find that the two sequences $\left(\mathbf{z}_{\boldsymbol{\lambda}}^{(2k)}\right)_{k\in\mathbb{N}}$ and $\left(\mathbf{z}_{\boldsymbol{\lambda}}^{(2k+1)}\right)_{k\in\mathbb{N}}$ converge to two limit points $\tilde{\mathbf{z}}_{\boldsymbol{\lambda},1}$ and $\tilde{\mathbf{z}}_{\boldsymbol{\lambda},2}$ for even and odd iterations respectively. Note that we do not assume here that $\tilde{\mathbf{z}}_{\boldsymbol{\lambda},1}$ and $\tilde{\mathbf{z}}_{\boldsymbol{\lambda},2}$ are equal. With these auxiliary limit points in mind, from the dual update equation in Algorithm 4 we know that

$$\boldsymbol{\lambda}^{(k+1)} = \mathbf{z}_{\boldsymbol{\lambda}}^{(k)} - \rho\left(\mathbf{Cx}^{(k+1)}-\mathbf{d}\right).$$

By substituting this relationship into (4.9), it follows that

$$\mathbf{z}_{\boldsymbol{\lambda}}^{(k+2)} - \mathbf{z}_{\boldsymbol{\lambda}}^{(k)} = -2\rho\left(\mathbf{P}\boldsymbol{\lambda}^{(k+2)}-\boldsymbol{\lambda}^{(k+1)}\right).$$

In the limit, we can therefore establish that

$$\lim_{k\to\infty}\left(\mathbf{z}_{\boldsymbol{\lambda}}^{(k+2)}-\mathbf{z}_{\boldsymbol{\lambda}}^{(k)}\right) = -2\rho\lim_{k\to\infty}\left(\mathbf{P}\boldsymbol{\lambda}^{(k+2)}-\boldsymbol{\lambda}^{(k+1)}\right) = -2\rho\left(\mathbf{P}\tilde{\boldsymbol{\lambda}}_2-\tilde{\boldsymbol{\lambda}}_1\right)=\mathbf{0}, \tag{4.10}$$

and thus that $\mathbf{P}\tilde{\boldsymbol{\lambda}}_2=\tilde{\boldsymbol{\lambda}}_1$ where $\tilde{\boldsymbol{\lambda}}_1=\tilde{\mathbf{z}}_{\boldsymbol{\lambda},1}-\rho\,(\mathbf{C}\tilde{\mathbf{x}}-\mathbf{d})$ for even iterations and $\tilde{\boldsymbol{\lambda}}_2=\tilde{\mathbf{z}}_{\boldsymbol{\lambda},2}-\rho\,(\mathbf{C}\tilde{\mathbf{x}}-\mathbf{d})$ for odd iterations. Given the existence of these dual limit states and the relationship between them, we are finally ready to prove the optimality of the primal iterates in the limit.

4.4.4. OPTIMALITY OF THE PRIMAL-DUAL LIMIT STATE

Given the properties developed in the previous sections, we are able to demonstrate that the primal variables of m-PDMM converge to a minimizer of (4.1). To achieve this point, we can adopt the simplified notation introduced in Section 4.3.1 to construct the equivalent problem given by

$$\min_{\mathbf{x}_i\,\forall\, i\in V} f(\mathbf{x}) \quad \text{s.t}\, \frac{1}{2}(\mathbf{PC}+\mathbf{C})\mathbf{x}=\mathbf{d}. \tag{4.11}$$

To prove that the vector $\tilde{\mathbf{x}}$ solves (4.1) we can therefore show that it equivalently solves (4.11). As (4.11) is convex, this can be achieved by considering its Lagrangian which is given by

$$\mathscr{L}(\mathbf{x},\boldsymbol{\mu}) = f(\mathbf{x}) - \frac{1}{2}\langle \boldsymbol{\mu}, (\mathbf{PC}+\mathbf{C})\mathbf{x}-2\mathbf{d}\rangle, \tag{4.12}$$

where the introduced vector $\boldsymbol{\mu}$ denotes the dual variables associated with the constraints in (4.11) and has the same dimension as the $\boldsymbol{\lambda}$ variables of the modified PDMM algorithm. As $f\in\Gamma^0$, it follows that the necessary and sufficient Karush-Kuhn-Tucker (KKT) conditions for solving (4.2) are given by

$$\text{Primal Feasibility:} \quad (\mathbf{PC}+\mathbf{C})\mathbf{x}^* = 2\mathbf{d}$$

$$\text{Subdifferential of Lagrangian:} \quad \mathbf{0}\in \frac{\partial \mathscr{L}(\mathbf{x}^*,\boldsymbol{\mu}^*)}{\partial \mathbf{x}} = \partial f(\mathbf{x}^*) - \frac{1}{2}(\mathbf{PC}+\mathbf{C})^T\boldsymbol{\mu}^*$$

If a limit states of the primal and dual variables satisfy these conditions then they also correspond to a primal-dual optimal point. Specifically, from Section 4.4.2, we already know that the point $\tilde{\mathbf{x}}$ is primal feasible. Therefore, if we can show that the points $(\tilde{\mathbf{x}},\tilde{\boldsymbol{\lambda}}_1)$ and $(\tilde{\mathbf{x}},\tilde{\boldsymbol{\lambda}}_2)$ both correspond to a zero of the subdifferential of (4.12) then they correspond to primal-dual optimal points of (4.11) and thus $\tilde{\mathbf{x}}$ is primal optimal. To demonstrate this, consider again the primal update equation given by

$$\mathbf{x}^{(k+1)} = \arg\min_{\mathbf{x}}\left(f(\mathbf{x}) - \left\langle \mathbf{C}^T\mathbf{z}_{\boldsymbol{\lambda}}^{(k)},\mathbf{x}\right\rangle + \frac{\rho}{2}\|\mathbf{Cx}-\mathbf{d}\|^2 + \frac{\gamma}{2}\left\|\mathbf{x}-\mathbf{x}^{(k)}\right\|^2\right),$$

where we have again used the fact that $\mathbf{z}_{\mathbf{x}}^{(k)} = \mathbf{x}^{(k)}$. By noting that $\mathbf{z}_{\boldsymbol{\lambda}}^{(k)} = \mathbf{P}\left(\boldsymbol{\lambda}^{(k)} - \rho\left(\mathbf{Cx}^{(k)}-\mathbf{d}\right)\right)$ it follows that

$$\mathbf{x}^{(k+1)} = \arg\min_{\mathbf{x}}\left(f(\mathbf{x}) - \left\langle \mathbf{C}^T\mathbf{P}\boldsymbol{\lambda}^{(k)},\mathbf{x}\right\rangle + \frac{\rho}{2}\left\|\mathbf{Cx}+\mathbf{PCx}^{(k)}-2\mathbf{d}\right\|^2 + \frac{\gamma}{2}\left\|\mathbf{x}-\mathbf{x}^{(k)}\right\|^2\right).$$

Assuming an even number of iterations, in the limit we have that

$$\tilde{\mathbf{x}} = \arg\min_{\mathbf{x}}\left(f(\mathbf{x}) - \langle \mathbf{CP}\tilde{\boldsymbol{\lambda}}_1,\mathbf{x}\rangle + \frac{\rho}{2}\|\mathbf{Cx}+\mathbf{PC}\tilde{\mathbf{x}}-2\mathbf{d}\|^2 + \frac{\gamma}{2}\|\mathbf{x}-\tilde{\mathbf{x}}\|^2\right),$$

and thus that

$$\mathbf{0}\in\partial f(\tilde{\mathbf{x}}) - \mathbf{CP}\tilde{\boldsymbol{\lambda}}_1 + \rho(\mathbf{C}\tilde{\mathbf{x}}+\mathbf{PC}\tilde{\mathbf{x}}-2\mathbf{d}) + \gamma(\tilde{\mathbf{x}}-\tilde{\mathbf{x}}).$$

Using the primal feasibility of $\tilde{\mathbf{x}}$, the fact that $\mathbf{P}\tilde{\boldsymbol{\lambda}}_1$ and cancelling common terms it follows that

$$\mathbf{0}\in\partial f(\tilde{\mathbf{x}}) - \mathbf{C}^T\mathbf{P}\tilde{\boldsymbol{\lambda}}_1 = \partial f(\tilde{\mathbf{x}}) - \mathbf{C}^T\tilde{\boldsymbol{\lambda}}_2.$$

Similarly, for an odd number of iterations, in the limit we have that

$$\mathbf{0} \in \partial f(\tilde{\mathbf{x}}) - \mathbf{C}^T \mathbf{P} \tilde{\boldsymbol{\lambda}}_2 = \partial f(\tilde{\mathbf{x}}) - \mathbf{C}^T \tilde{\boldsymbol{\lambda}}_1,$$

where we have made use of the result in (4.10). Combining these conditions, we find that

$$\mathbf{0} \in \partial f(\tilde{\mathbf{x}}) - \frac{1}{2}(\mathbf{PC}+\mathbf{C})^T \tilde{\boldsymbol{\lambda}}_1, \; \mathbf{0} \in \partial f(\tilde{\mathbf{x}}) - \frac{1}{2}(\mathbf{PC}+\mathbf{C})^T \tilde{\boldsymbol{\lambda}}_2$$

It follows that $(\tilde{\mathbf{x}}, \tilde{\boldsymbol{\lambda}}_1)$ and $(\tilde{\mathbf{x}}, \tilde{\boldsymbol{\lambda}}_2)$ are both primal-dual optimal points of (4.12) and thus that $\tilde{\mathbf{x}}$ minimizer of (4.11) and therefore (4.1). In this way, the m-PDMM algorithm is guaranteed to converge to an optimal primal point with only the use of primal averaging for all CCP functions.

4

4.5. NUMERICAL EXPERIMENTS

In this section, we verify the convergence of the m-PDMM algorithm for the case of a general CCP function. Specifically, we consider the counter example for PDMM demonstrated in Chapter 3, that of the L1 consensus problem given by

$$\min_{\mathbf{x}\in\mathbb{R}^N} \sum_{i\in V} \|\mathbf{x}_i - \mathbf{a}_i\|_1 \quad \text{s.t.}\, \mathbf{x}_i - \mathbf{x}_j = \mathbf{0} \; \forall (i,j) \in E. \tag{4.13}$$

For these simulations, we will restrict the dimension of each local variable to be of length five, i.e. $\mathbf{x}_i \in \mathbb{R}^5 \; \forall i \in V$. Additionally, the underlying network will be a $N = 51$ node Erdős-Rényi network. As in Chapter 3, the connection probability was selected as $\frac{\ln(N)}{N}$ to produce, with high probability, a connected network with few connections between nodes. Additionally the resulting network was ensured to form a single connected component as per our assumptions outlined in Section 4.3.1. Finally, step sizes of $\rho = \gamma = 1$ were selected. Note that these step sizes were by no means optimal and were selected purely for demonstration purposes. Incorporating the problem structure of (4.13) into the m-PDMM scheme, it follows that the iterations can be computed via Algorithm 5

Algorithm 5 Distributed L1 Consensus via m-PDMM

1: **Initialise:** $\mathbf{z}^{(0)} \in \mathbb{R}^{M_E}$
2: **for** k=0,..., **do**
3: $\quad \mathbf{x}^{(k+1)} = \underset{\mathbf{x}}{\operatorname{argmin}} \left(\|\mathbf{x}-\mathbf{a}\|_1 - \left\langle \mathbf{C}^T \mathbf{z}_{\boldsymbol{\lambda}}^{(k)}, \mathbf{x} \right\rangle + \frac{\rho}{2} \|\mathbf{Cx}\|^2 + \frac{\gamma}{2} \left\|\mathbf{x}-\mathbf{x}^{(k)}\right\|^2 \right)$
4: $\quad \mathbf{z}_{\boldsymbol{\lambda}}^{(k+1)} = \mathbf{P}\left(\mathbf{z}_{\boldsymbol{\lambda}}^{(k)} - 2\rho \mathbf{C}\mathbf{x}^{(k+1)}\right)$
5: **end for**

The convergence plot in Figure 4.1 demonstrates the squared primal objective gap $\|f(\mathbf{x}) - f\left(\mathbf{x}^{(k+1)}\right)\|^2$ with respect to the iteration number, i.e. the distance between the objective of the current iterates and that of a primal optimal solution. This metric was adopted due to the lack of uniqueness of the optimal solution for the proposed problem. For comparisons sake we have also included the standard PDMM algorithm to demonstrate its lack of convergence.

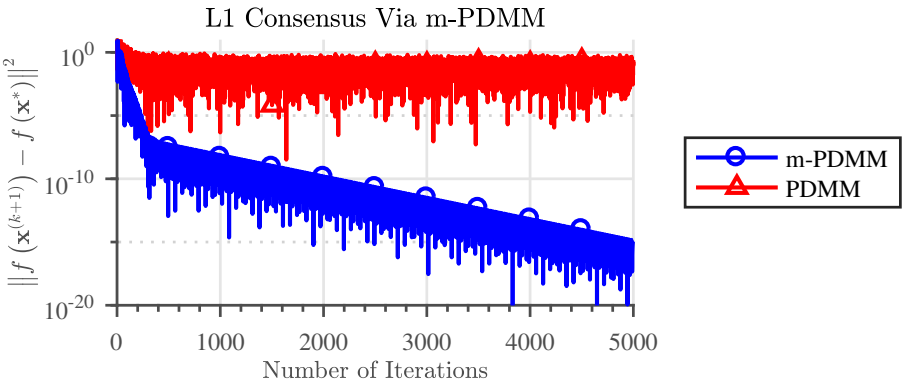


Figure 4.1: An example convergence plot of distributed L1 consensus via the m-PDMM and PDMM algorithms. While the PDMM algorithm fails to converge, the m-PDMM algorithm continues to converge towards a primal optimal solution.

In contrast to the standard PDMM algorithm, which we know cannot be guaranteed to converge for such problems, here we see that m-PDMM has no trouble in converging to an optimal solution. While being a simple example, this result reflects our analysis from the previous sections and shows that the m-PDMM approach allows us to address the lack of convergence of PDMM for a broader class of functions.

4.6. CONCLUSIONS

In this chapter we have presented a novel method of modifying the PDMM algorithm to guarantee convergence for all closed, convex and proper functions. Initially motivated by the manner in which PDMM discards the primal variables between iterations, in this work we have presented a method of incorporating this information through primal regularization. Notably, we demonstrated how this regularization can be incorporated through a primal-dual algorithmic formulation and the use of a partial averaging step over the primal variables at each node. Importantly, this allowed us to interpret the algorithm from the perspective of monotone operator theory and use results from this area to guarantee convergence. We further validated this point by demonstrating convergence in the case of a simple L1 consensus problem for which PDMM does not converge. Overall, the proposed method broadens the functional class able to be solved via a PDMM type approach without the use of a full averaging step and while preserving the desired distributed nature of the algorithm.

APPENDICES

4.A. PROOF OF LEMMA 4.3.1

To compute the reflected resolvent operator we will first begin by computing the resolvent operator

$$\mathbf{J}_{\rho,\mathbf{T}_1}\left(\begin{bmatrix}\mathbf{z}_{\boldsymbol{\lambda}}^{(k)}\\ \mathbf{z}_{\mathbf{x}}^{(k)}\end{bmatrix}\right)=\begin{bmatrix}\boldsymbol{\lambda}^{(k+1)}\\ \mathbf{x}^{(k+1)}\end{bmatrix}=\left(\mathbf{I}+\rho\mathbf{T}_1\right)^{-1}\begin{bmatrix}\mathbf{z}_{\boldsymbol{\lambda}}^{(k)}\\ \mathbf{z}_{\mathbf{x}}^{(k)}\end{bmatrix}.$$

By simply rearranging it follows that

$$\begin{bmatrix}\boldsymbol{\lambda}^{(k+1)}\\ \mathbf{x}^{(k+1)}\end{bmatrix}=\begin{bmatrix}\mathbf{z}_{\boldsymbol{\lambda}}^{(k)}\\ \mathbf{z}_{\mathbf{x}}^{(k)}\end{bmatrix}-\rho\mathbf{T}_1\left(\begin{bmatrix}\boldsymbol{\lambda}^{(k+1)}\\ \mathbf{x}^{(k+1)}\end{bmatrix}\right),$$

such that

$$\boldsymbol{\lambda}^{(k+1)}=\mathbf{z}_{\boldsymbol{\lambda}}^{(k)}-\rho\left(\mathbf{C}\mathbf{x}^{(k+1)}-\mathbf{d}\right),\ \mathbf{x}^{(k+1)}\in\mathbf{z}_{\mathbf{x}}^{(k)}-\rho\beta\left(\partial f\left(\mathbf{x}^{(k+1)}\right)-\rho\mathbf{C}^T\left(\mathbf{z}_{\boldsymbol{\lambda}}^{(k)}-\rho\left(\mathbf{C}\mathbf{x}^{(k+1)}-\mathbf{d}\right)\right)\right).$$

It follows that

$$0\in\partial f\left(\mathbf{x}^{(k+1)}\right)-\mathbf{C}^T\mathbf{z}_{\boldsymbol{\lambda}}^{(k)}+\rho\mathbf{C}^T\left(\mathbf{C}\mathbf{x}^{(k+1)}-\mathbf{d}\right)+\frac{1}{\rho\beta}\left(\mathbf{x}^{(k+1)}-\mathbf{z}_{\mathbf{x}}^{(k)}\right).$$

Given the inner product space determined by $\tilde{\mathscr{F}}$ it follows that

$$\mathbf{x}^{(k+1)}=\operatorname*{argmin}_{\mathbf{x}}\left(f(\mathbf{x})-\left\langle\mathbf{C}^T\mathbf{z}_{\boldsymbol{\lambda}}^{(k)},\mathbf{x}\right\rangle+\frac{\rho}{2}\left\|\mathbf{C}\mathbf{x}-\mathbf{d}\right\|^2+\frac{1}{2\rho\beta}\left\|\mathbf{x}-\mathbf{z}_{\mathbf{x}}^{(k)}\right\|^2\right).$$

Applying the definition of the reflected resolvent completes the proof. □

4.B. PROOF OF LEMMA 4.3.2

Again we will begin by exploiting the relationship between the reflected resolvent and resolvent operators where

$$\mathbf{J}_{\rho,\mathbf{T}_2}\left(\begin{bmatrix}\mathbf{y}_{\boldsymbol{\lambda}}^{(k+1)}\\ \mathbf{y}_{\mathbf{x}}^{(k+1)}\end{bmatrix}\right)=\left(\mathbf{I}+\rho\mathbf{T}_2\right)^{-1}\begin{bmatrix}\mathbf{y}_{\boldsymbol{\lambda}}^{(k+1)}\\ \mathbf{y}_{\mathbf{x}}^{(k+1)}\end{bmatrix}.$$

Immediately one can note that, due to the structure of $\mathbf{T}_2$ that

$$\mathbf{z}_{\mathbf{x}}^{(k+1)}=\mathbf{y}_{\mathbf{x}}^{(k+1)}.$$

In the case of $\mathbf{z}_{\boldsymbol{\lambda}}^{(k+1)}$, the computation of the resolvent is in fact a simple proximal step such that

$$\mathbf{z}_{\boldsymbol{\lambda}}^{(k+\frac{1}{2})} = \underset{(\mathbf{I}-\mathbf{P})\mathbf{u}=\mathbf{0}}{\arg\min} \left(\frac{1}{2} \left\| \mathbf{y}_{\boldsymbol{\lambda}}^{(k+1)} - \mathbf{u} \right\|^2 \right).$$

By observation, the resulting vector is given by $\mathbf{z}_{\boldsymbol{\lambda}}^{(k+\frac{1}{2})} = \frac{1}{2}(\mathbf{P}+\mathbf{I})\,\mathbf{y}_{\boldsymbol{\lambda}}^{(k+1)}$. By then applying the definition of the reflected resolvent it follows that $\mathbf{z}_{\boldsymbol{\lambda}}^{(k+1)} = \mathbf{P}\mathbf{y}_{\boldsymbol{\lambda}}^{(k+1)}$ completing the proof. □

5

NETWORK TOPOLOGY AND PDMM: CONVERGENCE RATE ANALYSIS

Thomas Sherson, Richard Heusdens, and W. Bastiaan Kleijn

"Your not going to find your epiphany moment in some holy place. You want a real epiphany, go stand on top of a mountain. Let the grandeur of it all wash over you and bask in the glory of your insignificance compared to those majestic peaks. Then and only then, when faced with the scale of it all can you reach a place of inner peace."

Anonymous

As our final contribution with regards to the analysis of existing distributed solvers, in this chapter we investigate the manner in which network topology impacts the convergence rates of PDMM. Specifically, we consider the case in which the objective functions are both strongly convex and smooth. We demonstrate a new convergence bound for this algorithm by linking its worst-case convergence with that of the methods of alternating projections and provide an analytic expression for the limiting convergence rate as the number of iterations increases. For consensus type problems we then show how this bound is related to the eigenvalues of the normalized adjacency matrix of the underlying network and thus with the mixing rate of random walks over its graph. Given this insight, we compare the effect of network topology on the convergence of PDMM for a range of different deterministic graphs and demonstrate convergence in finite iterations in the case of distributing averaging over a certain fully connected bipartite graph.

To the Norse gods and your thirteen hour storm, I have not forgotten that day *(July 5th, 2017, Lake Álftavatn, Iceland)*. Your contributions to this chapter, however unintentional, are greatly appreciated.

5.1. INTRODUCTION

The past three decades has seen an explosion in the dependency of services within our society on networking. From the advent of the internet, to the development of social media [1], the rise of the “Internet of Things" paradigm [5] and even the growth of blockchain based utilities[91], our society is becoming increasingly dependent on networked systems for everyday operation. However, as these systems grow in size and complexity, traditional signal processing approaches quickly become infeasible to implement due to the distribution of data within such networks and the scaling costs of data aggregation and centralization.

In response to this challenge, the last decade has seen a dramatic increase in the interest in and deployment of alternative methods of signal processing based on distributed computation. Such methods can take a variety of forms including the likes of distributed consensus [10, 12, 72, 92, 93], belief propagation/message passing approaches [15, 14, 13, 94], graph signal processing over networks[73, 95, 17, 74], distributed optimization [18, 96, 97, 19, 79] and more. The choice between these different approaches is made based on the specific problems to be solved and often trades between complexity and convergence rate.

An important question that arises for all distributed methods is how the underlying topology of a network impacts their rate of convergence. In the case of distributed averaging and graph filtering type approaches, the underlying linearity of the operators involved allows for an analysis of this point [98, 17]. For instance, in the case of distributed averaging, a connection can be readily drawn between convergence rate and the structure of the averaging operators involved, notably that the extremal eigenvalues of such operators characterize their rate. However, for other approaches, such as those based on message passing and distributed optimization, this question remains an open problem.

The work in this article aims to address this question for a specific distributed optimization algorithm recently proposed within the literature termed the primal dual method of multipliers (PDMM) [96]. Following its initial introduction, this method was shown to be equivalent to an instance of Peaceman-Rachford splitting applied to a certain lifted dual problem [99] in which geometric convergence was also demonstrated for strongly convex, smooth functions. Furthermore, it was shown in [96] that PDMM performs favorably compared to other distributed methods such as the alternating direction method of multipliers (ADMM). Motivated by these findings, within this work a connection is made between the convergence of PDMM for strongly convex, smooth functions and the generalized alternating method of projections. Using this insight, a novel iteration-tight convergence bound for the worst-case convergence of PDMM is demonstrated and, more importantly, the rate of this bound is linked with the underlying structure of the distributed network itself.

5.1.1. RELATED WORK

At its heart, this paper builds on efforts within the literature on distributed optimization developed over the past four decades. While initial efforts within the literature began in the 1980’s and early 1990’s in the context of parallel computing[29, 30, 31, 32, 33], distributed optimization has seen a resurgence in popularity in recent year for use in a vari-

ety of applications including environmental monitoring [100], multi-agent coordination [101] and more.

The specific task of considering the effect of network topology on distributed algorithms has also been raised in recent years as an important aspect of algorithm selection (see [102] for a recent paper on this point). For different algorithms, this analysis has taken on various forms but the need to understand the role network design plays is nonetheless prevalent.

In [98] for instance, the problem of designing the transition matrix resulting in the fastest convergence of a distributed averaging problem was considered for a given network topology. Such an approach provides a lower bound on the fastest rates of convergence that can be achieved for a given topology and in turn provides indirect insight into the role network structure plays in algorithmic performance.

In the case of distributed graph filtering such as that in [17], the effect of network topology is also considered. In particular, the distributed filters designed therein utilize the Laplacian of the underlying graphical model as a kernel for ARMA style filters. The convergence characteristics of these filters are in turn determined by the spectrum of said Laplacians, as noted in the proof of [17, Proposition 2] and thus provide a clear link between algorithm performance and network topology.

The main aspect that enables the analysis of the approaches mentioned above is the underlying linearity of the algorithm updates. In the case of more general classes of distributed solvers, such approaches are less applicable. However, for specific algorithms, a number of interesting results do exist. For instance, the work of [103] examined the case of a proposed dual distributed averaging subgradient algorithm and drew an explicit link between the underlying network topology and the convergence of the approach. It was shown that for networks with favorable connectivity, faster convergence rates could be guaranteed. Such a result reflects similar findings in the field of spectral graph theory and highlights the importance of good network design in such a context.

The work of [104] examined the rate of convergence of a distributed instance of the alternating direction method of multipliers (ADMM). A novel convergence proof was demonstrated for the algorithm which again provides an explicit link with network topology. In particular, the geometric convergence rate of ADMM for twice differentiable functions was quantified and shown to be directly related to the underlying topology. In the specific case of a star and a ring network, this rate was made explicit allowing a comparison of their performance.

The effect of network topology on ADMM was also considered in [105] where a connection was drawn between lifted Markov chains and ADMM. This work was extended in the case of [106] to explore the overall affects of network topology on the performance of over-relaxed ADMM. In particular, for a certain non-strongly convex quadratic problem, the convergence of over-relaxed ADMM for a given network was quantified by exploiting the aforementioned connection with lifted Markov chains. However, more general problem classes were not considered therein.

More recently, the work of [102] provided an overview of distributed averaging algorithms and a distributed subgradient generalization of such approaches. Within this discussion, the authors provide insight into the role that network topology plays in such methods but an explicit quantification of the effect on convergence rate is not provided

in all circumstances. However, a particular point where this is addressed is in [102, Corollaries 5 and 9] for the case of distributed averaging over undirected networks and distributed convex consensus over the same networks. While only a certain set of graph topologies were considered, the approach provides insight into the tradeoff between different network structures and overall algorithmic convergence.

5.1.2. MAIN CONTRIBUTIONS

The contributions of this paper are three-fold. Motivated by the promising performance of PDMM for distributed optimization, we firstly demonstrate a new tighter bound for its convergence for smooth, strongly convex functions. This bound strictly improves upon the geometric convergence rate demonstrated in [99] for the same functional class. We additionally extend this result as the number of iterations tends to infinity by defining the limiting worst-case rate for PDMM.

Secondly, for consensus type problems, an explicit link between network topology and convergence rate is highlighted. In particular, we demonstrate how PDMM is parameterized by the second largest absolute eigenvalue of the random walk matrix of the network. In this way, we directly show that the mixing characteristics of the network determine the overall convergence of PDMM for the considered problem class.

Finally we consider the role of network structure on the worst-case convergence rate of PDMM for a number of specific graph topologies including chain, lattice, grid, bipartite and fully connected graphs. This allows us to compare the performance of PDMM for these structures and highlight the importance of good network design in a distributed optimization context. We also demonstrate a specific problem instance where PDMM converges in finite iterations.

5.1.3. ORGANIZATION OF PAPER

The remainder of the paper is constructed as follows. In Section 5.2 we introduce appropriate nomenclature to support the remainder of this document. In Section 5.3 we briefly outline the PDMM algorithm and its associated notation. Section 5.4 introduces a new convergence bound for PDMM applied to strongly convex, smooth functions while Section 5.5 complements this with a set of additional convergence results including a problem instance which attains the worst-case convergence rate and a method for optimal step-size selection. Section 5.6 demonstrates the impact network topology has on this bound in the case of consensus type problems and demonstrates this effect for specific graph topologies. Finally, in Section 5.7 we draw our conclusions on the paper.

5.2. NOMENCLATURE

We denote by $\mathbb{R}$ the set of real numbers, by $\mathbb{R}^N$ the set of real column vectors of length N and by $\mathbb{R}^{M\times N}$ the set of M by N real matrices. Similarly, regular lowercase letters or symbols will be used to denote scalar values, i.e., x while bold font lower case will be used to denote vectors, i.e., $\mathbf{x}$. Matrices will be denoted by bold font uppercase letters or symbols, i.e., $\mathbf{X}$. Given a matrix $\mathscr{A}$, the orthogonal projection onto the range of $\mathscr{A}$ is given by $\underset{\mathscr{A}}{\Pi} = \mathscr{A}\left(\mathscr{A}^T\mathscr{A}\right)^{\dagger}\mathscr{A}^T$ where $\dagger$ denotes the Moore-Penrose pseudo inverse. Finally, ∇f denotes the gradient of a differentiable function f.

5.3. DISTRIBUTED OPTIMIZATION VIA THE PRIMAL DUAL METHOD OF MULTIPLIERS

In this section we summarize a recently proposed algorithm for distributed optimization termed the primal dual method of multipliers (PDMM). The original treatment of this algorithm can be found in [96] while a more recent publication, which approaches the analysis of PDMM from the perspective of monotone operator theory is given in [99].

5.3.1. PROBLEM DEFINITION

Consider a simple, undirected, N node network $G(V,E)$ parameterized by node/vertex set $V = \{1,2,\cdots,N\}$ and an undirected edge set $E = \{(i,j)\}$ where $(i,j) \in E$ if nodes i and j can communicate. Our objective it to use such a network to perform distributed convex optimization. Assume that each node $i \in V$ is equipped with a local function $f_i \in \Gamma_0\left(\mathbb{R}^{M_i}\right)$ parameterized by $\mathbf{x}_i \in \mathbb{R}^{M_i}$. Here Γ_0 denotes the family of closed, convex and proper (CCP) functions. Given such functions we will consider solving problems of the form

$$\min_{\mathbf{x}_i\, \forall\, i \in V} \sum_{i \in V} f_i(\mathbf{x}_i) \quad \text{s.t}\, \mathbf{A}_{i|j}\mathbf{x}_i + \mathbf{A}_{j|i}\mathbf{x}_j = \mathbf{b}_{i,j} \quad \forall\, (i,j) \in E, \tag{5.1}$$

where the terms $\mathbf{A}_{i|j} \in \mathbb{R}^{M_{i,j} \times M_i}$ and $\mathbf{b}_{i,j} \in \mathbb{R}^{M_{i,j}}$ impose linear constraints between neighboring nodes, the identifier $i|j$ denotes a directed edge from i to j while i,j denotes an undirected edge. Furthermore, let $M_V = \sum_{i \in V} M_i$ and $M_E = \sum_{(i,j) \in E} M_{i,j}$ and assume that (5.1) is feasible. Importantly, (5.1) is a separable problem with the objective function divided amongst the nodes and the constraints only acting over the physical edges of the network. It is this fundamental structure which allows us to optimize (5.1) in a distributed manner.

5.3.2. SIMPLIFICATION OF NOTATION

To assist in the introduction of PDMM, we define a compact vector notation to simplify the considered prototype problem. Specifically, we show that (5.1) can be rewritten as

$$\min_{\mathbf{x}} f(\mathbf{x}) \quad \text{s.t}\, \mathbf{C}\mathbf{x} + \mathbf{P}\mathbf{C}\mathbf{x} = 2\mathbf{d}, \tag{5.2}$$

where $\mathbf{C}$, $\mathbf{P}$ and $\mathbf{d}$ are defined in the following.

COMPACT OBJECTIVE NOTATION

To parameterize the simplified problem we firstly introduce the stacked vector variable

$$\mathbf{x} = [\mathbf{x}_1^T, \mathbf{x}_2^T, \cdots, \mathbf{x}_N^T]^T.$$

Using this vector notation, the objective function can be simplified by defining the global function

$$f : \mathbb{R}^{M_V} \mapsto \mathbb{R},\ \mathbf{x} \mapsto \sum_{i \in V} f_i(\mathbf{x}_i)$$

as the sum of all local functions.

COMPACT CONSTRAINTS NOTATION

Using the same vector notation, we can form a compact representation for the constraint functions of (5.1). To do so, we define the additional matrix $\mathbf{C} \in \mathbb{R}^{M_E \times M_V}$ and vector $\mathbf{d} \in \mathbb{R}^{M_E}$. In the case of a fully connected network, these terms are given by

$$\mathbf{C}_i = \left[\mathbf{A}_{i|1}^T, \cdots, \mathbf{A}_{i|i-1}^T, \mathbf{A}_{i|i+1}^T, \cdots, \mathbf{A}_{i|N}^T\right]^T, \ \mathbf{d}_i = \frac{1}{2}\left[\mathbf{b}_{i,1}^T, \cdots, \mathbf{b}_{i,i-1}^T, \mathbf{b}_{i,i+1}^T, \cdots, \mathbf{b}_{i,N}^T\right]^T \ \forall i \in V,$$

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_1 & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{C}_N \end{bmatrix}, \ \mathbf{d} = \left[\mathbf{d}_1^T, \cdots, \mathbf{d}_N^T\right]^T. \tag{5.3}$$

For other network topologies the unnecessary rows of $\mathbf{C}$ and $\mathbf{d}$, corresponding to non-existent edges, can be removed.

Given the parameterization of $\mathbf{C}$ and $\mathbf{d}$ by the underlying constraint matrices, $\mathbf{A}_{i|j}$ and $\mathbf{b}_{i,j}$, all that remains is to define the symmetric permutation matrix $\mathbf{P}$ to ensure that the constraints of (5.1) correspond to

$$\mathbf{Cx} + \mathbf{PCx} = 2\mathbf{d}.$$

To define $\mathbf{P}$, note that the particular structure of $\mathbf{C}$ and $\mathbf{d}$ imposes an ordering on the directed edge index $i|j$ where $1|2 < 1|3 < \cdots < 1|N < 2|1 < 2|3 < \cdots < N|N-1$. It follows that $\mathbf{P}$ is a symmetric permutation matrix given by

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{1|2,1|2} & \cdots & \mathbf{P}_{1|2,N|N-1} \\ \cdots & \ddots & \cdots \\ \mathbf{P}_{N|N-1,1|2} & \cdots & \mathbf{P}_{N|N-1,N|N-1} \end{bmatrix}$$

where each block is defined as

$$\mathbf{P}_{i|j,k|l} = \begin{cases} \mathbf{I}_{M_{i,j}} & \text{if } i = l, j = k \\ \mathbf{0}_{M_{i,j} \times M_{k,l}} & \text{otherwise} \end{cases}$$

5.3.3. PDMM ALGORITHM

Given the compact notation demonstrated above, it was shown in [99] that the simplified problem in (5.2) can be optimized via PDMM iteratively for a given step size $\rho \in (0, +\infty)$ as shown in Algorithm 6. The PDMM algorithm itself is in fact an instance of Peaceman-Rachford splitting of a certain lifted dual form of (5.2).

As previously mentioned, PDMM reduces the global optimization of (5.1) into a sequence of parallelizable operations. The *primal updates*, corresponding to the *primal variables* $\mathbf{x}^{(k+1)}$ and $\mathbf{y}^{(k+1)}$, are computed in parallel across all nodes due to the block diagonal structure of $\mathbf{C}$ and separability of f. In contrast, the permutation $\mathbf{P}$, associated with the *auxiliary updates* of the *auxiliary variables* $\mathbf{z}^{(k+1)}$, corresponds to an exchange of information along the edges of the network.

Algorithm 6 Primal-Dual Method of Multipliers (PDMM)

1: **Initialise:** $\mathbf{z}^{(0)} \in \mathbb{R}^{2M_E}$
2: **for** k=0,..., **do**
3: $\quad \mathbf{x}^{(k+1)} = \underset{\mathbf{x}}{\operatorname{argmin}} \left(f(\mathbf{x}) - \left\langle \mathbf{C}^T \mathbf{z}^{(k)}, \mathbf{x} \right\rangle + \frac{\rho}{2} ||\mathbf{C}\mathbf{x} - \mathbf{d}||^2 \right)$
4: $\quad \mathbf{y}^{(k+1)} = \mathbf{z}^{(k)} - 2\rho \left(\mathbf{C}\mathbf{x}^{(k+1)} - \mathbf{d} \right)$
5: $\quad \mathbf{z}^{(k+1)} = \mathbf{P}\mathbf{y}^{(k+1)}$
6: **end for**

5.4. A TIGHT GEOMETRIC CONVERGENCE BOUND FOR PDMM FOR STRONGLY CONVEX, SMOOTH FUNCTIONS

To demonstrate the relationship between the convergence rate of PDMM and the underlying network topology, we first need to understand the convergence characteristics of PDMM itself. For distributed problems with strongly convex and smooth objective functions, the fastest known convergence bound was demonstrated in Chapter 3 by linking the worst-case rate of PDMM with the alternating method of projections. In this section we strengthen this result by introducing a stronger convergence rate bound for PDMM as well as demonstrating its limiting rate as the number of iterations increases. In particular we highlight the interplay between the local updates at each node and the mixing effect that edge based exchanges between nodes has on overall convergence. In the case of consensus problems, in Section 5.6 we demonstrate how this mixing effect is directly parameterized by the random walk matrix of the underlying network.

5.4.1. PRELIMINARY FUNCTIONAL ASSUMPTIONS

The following section demonstrates a new bound on the *auxiliary error* $\left\| \mathbf{z}^{(k)} - \mathbf{z}^* \right\|^2$ where the sequence $\left(\mathbf{z}^{(k)} \right)_{k \in \mathbb{N}}$ is computed as per Algorithm 6 and $\mathbf{z}^*$ denotes an optimal instance of the auxiliary variables. Such a $\mathbf{z}^*$ produces a primal optimal update $\mathbf{x}^*$ which solves (5.1). We also demonstrate that the proposed bound is stronger than that in [99].

For the formation of the aforementioned auxiliary error bound we make use of the relationship between convexity of a function and the monotonicity of its subdifferential. Specifically, monotonicity of an operator is defined as follows.

Definition 5.4.1. *Monotone Operator: An operator* $\mathbf{T}$ *is monotone if for all* $(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2) \in \text{gra}(\mathbf{T})$

$$\left\langle \mathbf{y}_1 - \mathbf{y}_2, \mathbf{x}_1 - \mathbf{x}_2 \right\rangle \geq 0.$$

If a convex function is subdifferentiable then its subdifferential is a monotone operator. In addition to convexity, in the following we assume that f is μ-strongly convex and β-smooth. These properties are defined as follows.

Definition 5.4.2. *Strong Convexity: A function* f *is* μ*-strongly convex with* $\mu > 0$ *if* $\forall \mathbf{x}_1, \mathbf{x}_2 \in dom(f), \mathbf{y}_2 \in \nabla f(\mathbf{x}_2)$,

$$f(\mathbf{x}_1) \geq f(\mathbf{x}_2) + \left\langle \mathbf{y}_2, \mathbf{x}_1 - \mathbf{x}_2 \right\rangle + \frac{\mu}{2} \left\| \mathbf{x}_1 - \mathbf{x}_2 \right\|^2.$$

This implies that, $f - \frac{\mu}{2}\|\bullet\|^2$ is convex.

If f is μ-strongly convex, ∇f is μ-strongly monotone.

Definition 5.4.3. *Strong Monotonicity: The operator ∇f is μ-strongly monotone with $\mu > 0$, if $\forall \mathbf{x}_1, \mathbf{x}_2 \in dom(f), \mathbf{y}_1 \in \nabla f(\mathbf{x}_1), \mathbf{y}_2 \in \nabla f(\mathbf{x}_2)$,*

$$\langle \mathbf{y}_1 - \mathbf{y}_2, \mathbf{x}_1 - \mathbf{x}_2 \rangle \geq \mu \|\mathbf{x}_1 - \mathbf{x}_2\|^2.$$

Definition 5.4.4. *Smoothness: A convex function f is β-smooth with $\beta > 0$ if it is differentiable and $\forall \mathbf{x}_1, \mathbf{x}_2 \in dom(f)$,*

$$f(\mathbf{x}_1) \leq f(\mathbf{x}_2) + \langle \nabla f(\mathbf{x}_2), \mathbf{x}_1 - \mathbf{x}_2 \rangle + \frac{\beta}{2}\|\mathbf{x}_1 - \mathbf{x}_2\|^2.$$

This implies that, $\frac{\beta}{2}\|\bullet\|^2 - f$ is convex.

If f is β-smooth, ∇f is $\frac{1}{\beta}$-cocoercive.

Definition 5.4.5. *Cocoercive: The monotone operator ∇f is $\frac{1}{\beta}$-cocoercive with $\beta > 0$ if $\forall \mathbf{x}_1, \mathbf{x}_2 \in dom(f)$,*

$$\langle \nabla f(\mathbf{x}_1) - \nabla f(\mathbf{x}_2), \mathbf{x}_1 - \mathbf{x}_2 \rangle \geq \frac{1}{\beta}\|\nabla f(\mathbf{x}_1) - \nabla f(\mathbf{x}_2)\|^2.$$

Furthermore, as $f \in \Gamma_0$ and differentiable, if f is β-smooth, ∇f is β-Lipschitz continuous.

Definition 5.4.6. *Lipschitz Continuous: The operator ∇f is β-Lipschitz if $\forall \mathbf{x}_1, \mathbf{x}_2 \in dom(f)$,*

$$\|\nabla f(\mathbf{x}_1) - \nabla f(\mathbf{x}_2)\| \leq \beta \|\mathbf{x}_1 - \mathbf{x}_2\|.$$

If $\beta = 1$, ∇f is nonexpansive while if $\beta < 1$ it is contractive.

Additionally, many of the arguments used in this section make use of the notions of the kernel and range space of non-square matrices. These properties are defined below.

Definition 5.4.7. *Range and Kernel Space: Given a matrix $\mathbf{A} \in \mathbb{R}^{N \times M}$, the range of $\mathbf{A}$ is denoted by* $\text{ran}(\mathbf{A})$ *where*

$$\forall \mathbf{y} \in \text{ran}(\mathbf{A}), \exists \mathbf{u} \mid \mathbf{A}\mathbf{u} = \mathbf{y}.$$

Similarly, the kernel of $\mathbf{A}$ is denoted by $\ker(\mathbf{A})$ *where*

$$\forall \mathbf{y} \in \ker(\mathbf{A}), \mathbf{A}\mathbf{y} = \mathbf{0}.$$

For any matrix, the subspaces $\text{ran}(\mathbf{A})$ *and* $\ker(\mathbf{A}^T)$ *are orthogonal and their direct sum* $\text{ran}(\mathbf{A}) + \ker(\mathbf{A}^T) = \mathbb{R}^N$.

Under the assumption that f is μ-strongly convex and β-smooth, we are now ready to proceed with demonstrating our new convergence rate bound for PDMM.

5.4.2. INDEPENDENCE OF A NON-CONTRACTIVE SUBSPACE

As a combination of nonexpansive operators, the auxiliary error sequence of PDMM $\left(\left\|\mathbf{z}^{(k)}-\mathbf{z}^*\right\|^2\right)_{k\in\mathbb{N}}$ is guaranteed to be nonincreasing. In general however, this is not sufficient to guarantee convergence to an optimal $\mathbf{z}^*$, even for the functional class we consider. This point was previously noted in [99] and is reproduced here for completeness. By considering the PDMM updates as defined in Algorithm 6 we find that for two successive iterations of PDMM,

$$\mathbf{z}^{(k+1)}=\mathbf{P}\left(\mathbf{z}^{(k)}-2\rho\left(\mathbf{C}\mathbf{x}^{(k+1)}-\mathbf{d}\right)\right),\ \mathbf{z}^{(k+2)}=\mathbf{z}^{(k)}-2\rho\left(\mathbf{P}\mathbf{C}\mathbf{x}^{(k+2)}+\mathbf{C}\mathbf{x}^{(k+1)}-2\mathbf{d}\right).$$

From the feasibility of (5.1) and the definitions in Section 5.3.2, $\exists\mathbf{x}^*\mid\mathbf{P}\mathbf{C}\mathbf{x}^*+\mathbf{C}\mathbf{x}^*=2\mathbf{d}$ and thus that $\mathbf{d}\in\operatorname{ran}(\mathbf{P}\mathbf{C})+\operatorname{ran}(\mathbf{C})$. Every two PDMM updates can therefore only affect the auxiliary variables in the subspace $\operatorname{ran}(\mathbf{P}\mathbf{C})+\operatorname{ran}(\mathbf{C})$ while the component of $\mathbf{z}$ in $\ker\left(\mathbf{C}^T\right)\cap\ker\left(\mathbf{C}^T\mathbf{P}\right)$ is unaffected. It follows that the auxiliary variables cannot be guaranteed to converge in general as any initial vector $\mathbf{z}^{(0}$ may contain a component in this nondecreasing subspace.

From Algorithm 6, we can note that $\mathbf{x}^{(k+1)}$ is only dependent on $\underset{\operatorname{ran}(\mathbf{C})}{\Pi}\left(\mathbf{z}^{(k)}+\rho\mathbf{d}\right)$. As we are interested in the ability of PDMM to compute the primal minimizers of (5.1), we can therefore consider the convergence of the auxiliary variables within our domain of interest, i.e. the norm term

$$\left\|\underset{\operatorname{ran}(\mathbf{C})}{\Pi}\left(\mathbf{z}^{(k+1)}-\mathbf{z}^*\right)\right\|^2. \tag{5.4}$$

In this work, we approach the task of bounding (5.4) by utilizing the fact that $\ker\left(\mathbf{C}^T\right)\cap\ker\left(\mathbf{C}^T\mathbf{P}\right)\subseteq\ker\left(\mathbf{C}^T\right)$. It follows that by bounding the right hand side of the inequality

$$\left\|\underset{\operatorname{ran}(\mathbf{C})}{\Pi}\left(\mathbf{z}^{(k+1)}-\mathbf{z}^*\right)\right\|^2\leq\left\|\underset{\operatorname{ran}(\mathbf{P}\mathbf{C})+\operatorname{ran}(\mathbf{C})}{\Pi}\left(\mathbf{z}^{(k+1)}-\mathbf{z}^*\right)\right\|^2,$$

we can define an upper bound for (5.4) in turn. We shall refer to this as the *projected auxiliary error* from here on out. In the following subsections, the analysis of this projected error is broken down into two stages, relating to the node based $\mathbf{x}^{(k+1)}$ and $\mathbf{y}^{(k+1)}$ updates, and the edge based update of $\mathbf{z}^{(k+1)}$.

5.4.3. BOUNDING THE PRIMAL ERROR $\mathbf{y}^{(k+1)}-\mathbf{y}^*$

To form our convergence bound for PDMM, we begin by considering the effect of the primal update equations. As per Algorithm 6, these updates are given by

$$\begin{aligned}\mathbf{x}^{(k+1)}&=\underset{\mathbf{x}}{\operatorname{argmin}}\left(f(\mathbf{x})-\left\langle\mathbf{C}^T\mathbf{z}^{(k)},\mathbf{x}\right\rangle+\frac{\rho}{2}||\mathbf{C}\mathbf{x}-\mathbf{d}||^2\right)\\ \mathbf{y}^{(k+1)}&=\mathbf{z}^{(k)}-2\rho\left(\mathbf{C}\mathbf{x}^{(k+1)}-\mathbf{d}\right).\end{aligned} \tag{5.5}$$

By using the functional assumptions of f outlined in Section 5.4.1, in the coming subsections we show that, given an initial vector $\mathbf{z}^{(k)}$ and an optimal auxiliary point $\mathbf{z}^*$, the

updates $\mathbf{y}^{(k+1)}$ and $\mathbf{y}^* = \mathbf{z}^* - 2\rho\,(\mathbf{C}\mathbf{x}^* - \mathbf{d})$ satisfy the ellipsoid bound given by

$$\left\| \prod_{\text{ran}(\mathbf{C})} \left(\mathbf{y}^{(k+1)} - \mathbf{y}^* - (1-\hat{\beta}-\hat{\mu}) \left(\mathbf{z}^{(k)} - \mathbf{z}^* \right) \right) \right\| \leq (\hat{\beta}-\hat{\mu}) \left\| \prod_{\text{ran}(\mathbf{C})} \left(\mathbf{z}^{(k)} - \mathbf{z}^* \right) \right\|, \tag{5.6a}$$

$$\prod_{\ker(\mathbf{C}^T)} \left(\mathbf{y}^{(k+1)} - \mathbf{y}^* \right) = \prod_{\ker(\mathbf{C}^T)} \left(\mathbf{z}^{(k)} - \mathbf{z}^* \right), \tag{5.6b}$$

where the terms $\hat{\beta}$ and $\hat{\mu}$ are defined in Lemma 5.4.1.

The equality (5.6b) can be straightforwardly observed from the update equations given in (5.5) by noting that $\mathbf{x}^{(k+1)}$ can only influence those component of $\mathbf{y}^{(k+1)}$ in ran $(\mathbf{C})$. This leaves the component $\mathbf{y}^{(k+1)}$ contained within $\ker\left(\mathbf{C}^T\right)$ unaffected. In contrast, (5.6a) requires a more careful analysis as provided in the following two subsections.

5.4.4. PRESERVATION OF STRONG CONVEXITY AND SMOOTHNESS

To prove that (5.6a) holds, we first need to examine how the functional assumptions on f propagate through to the $\mathbf{y}^{(k+1)}$ updates. To this end, consider again the $\mathbf{x}^{(k+1)}$ update of PDMM given by

$$\mathbf{x}^{(k+1)} = \underset{\mathbf{x}}{\text{argmin}} \left(f(\mathbf{x}) - \left\langle \mathbf{C}^T \mathbf{z}^{(k)}, \mathbf{x} \right\rangle + \frac{\rho}{2} ||\mathbf{C}\mathbf{x} - \mathbf{d}||^2 \right). \tag{5.7}$$

As f is smooth and strongly convex, it follows that both $\mathbf{x}^{(k+1)}$ and $\nabla f(\mathbf{x}^{(k+1)})$ are unique such that

$$\left(\nabla f + \rho \mathbf{C}^T \mathbf{C} \right) \mathbf{x}^{(k+1)} = \mathbf{C}^T \left(\mathbf{z}^{(k)} + \rho \mathbf{d} \right), \tag{5.8}$$

where ∇f denotes the gradient written as an operator. By defining the set $\mathbf{X} = \left\{ \mathbf{x}^{(k+1)} \mid \exists \mathbf{z}^{(k)} \right.$ so that (5.7) holds$\left.\right\}$, it follows from (5.8) that, $\forall \mathbf{x} \in \mathbf{X}$, $\nabla f(\mathbf{x}) \in \text{ran}(\mathbf{C}^T)$.

Reconsidering (5.7), we can note that neither the inner product $\left\langle \mathbf{C}^T \mathbf{z}^{(k)}, \mathbf{x} \right\rangle$ nor the quadratic form $\frac{\rho}{2}||\mathbf{C}\mathbf{x} - \mathbf{d}||^2$ has any dependence on the component $\mathbf{x}_K = \prod_{\ker(\mathbf{C})} \mathbf{x}$. By defining the function g such that $g(\mathbf{x}_R) = \min_{\mathbf{x}_K \in \ker(\mathbf{C})} f(\mathbf{x}_R + \mathbf{x}_K)$ where $\text{dom}(g) = \left\{ \mathbf{x}_R \mid \mathbf{x} \in \right.$ $\left. \text{dom}\left(f\right), \mathbf{x}_R = \prod_{\text{ran}(\mathbf{C}^T)} \mathbf{x} \right\}$, it follows that $\mathbf{x}_R^{(k+1)} = \prod_{\text{ran}(\mathbf{C}^T)} \mathbf{x}^{(k+1)}$ can be computed as

$$\mathbf{x}_R^{(k+1)} = \underset{\mathbf{x}_R \in \text{ran}(\mathbf{C}^T)}{\text{argmin}} \left(g(\mathbf{x}_R) - \left\langle \mathbf{C}^T \mathbf{z}^{(k)}, \mathbf{x}_R \right\rangle + \frac{\rho}{2} ||\mathbf{C}\mathbf{x}_R - \mathbf{d}||^2 \right).$$

The function g can be thought of as the partial optimization of f over $\ker(\mathbf{C})$ for a given instance of $\mathbf{x}_R$. As in the case of (5.8), using the operator ∇g, where differentiability follows from that of f, we find that

$$\left(\nabla g + \rho \mathbf{C}^T \mathbf{C} \right) \mathbf{x}_R^{(k+1)} = \mathbf{C}^T \left(\mathbf{z}^{(k)} + \rho \mathbf{d} \right), \tag{5.9}$$

and thus that $\forall \mathbf{x} \in \mathbf{X}, \mathbf{x}_R = \prod_{\text{ran}(\mathbf{C}^T)} \mathbf{x}$

$$\nabla g\left(\mathbf{x}_R\right) = \nabla f\left(\mathbf{x}\right) \in \text{ran}\left(\mathbf{C}^T\right), \tag{5.10}$$

Using the definitions of $\mathbf{X}$ and $\mathbf{X}_R$, in conjunction with (5.10) allows us to demonstrate that, g inherits the smoothness and strong convexity of f $\forall \mathbf{x}_R \in \mathbf{X}_R$. This is summarized in the following two Propositions.

Proposition 5.4.1. *If* ∇f *is* $\frac{1}{\beta}$*-cocoercive, then* $\forall \mathbf{x}_a, \mathbf{x}_b \in \mathbf{X}_R$,

$$\left\langle \nabla g(\mathbf{x}_a) - \nabla g(\mathbf{x}_b), \mathbf{x}_a - \mathbf{x}_b \right\rangle \geq \frac{1}{\beta} \left\| \nabla g(\mathbf{x}_a) - \nabla g(\mathbf{x}_b) \right\|^2,$$

i.e., ∇g *is* $\frac{1}{\beta}$*-cocoercive with respect to* $\mathbf{X}_R$.

Proposition 5.4.2. *If* ∇f *is* μ*-strongly monotone then* $\forall \mathbf{x}_a, \mathbf{x}_b \in \mathbf{X}_R$,

$$\left\langle \nabla g(\mathbf{x}_a) - \nabla g(\mathbf{x}_b), \mathbf{x}_a - \mathbf{x}_b \right\rangle \geq \mu \left\| \mathbf{x}_a - \mathbf{x}_b \right\|^2,$$

i.e.,∇g *is* μ*-strongly monotone with respect to* $\mathbf{X}_R$.

The proofs of these Propositions can be found in Appendix 5.A and 5.B respectively.. From the link between strong monotonicity, strong convexity, cocoercivity and smoothness, it follows that g is μ-strongly convex and β-smooth with respect to $\mathbf{X}_R$.

5.4.5. FORMING THE ELLIPSOIDAL BOUND

Given the properties of g, we are now ready to form the ellipsoidal bound for $\left\| \Pi_{\text{ran}(\mathbf{PC})+\text{ran}(\mathbf{C})} \left(\mathbf{y}^{(k+1)} - \mathbf{y}^* \right) \right\|^2$ given in Eq. (5.6). The approach adopted in this section reflects the similar result demonstrated in [82] for the case of centralized optimization. Note that the results therein cannot be guaranteed to hold in the case of PDMM due to the potential violation of [82, Assumption 2].

Making use of the domain of ∇g the left hand side of (5.9) is equivalent to

$$\mathbf{C}^T \left(\rho \frac{\mathbf{C}^{\dagger T}}{\rho} \nabla g \frac{\mathbf{C}^{\dagger}}{\rho} + \mathbf{I} \right) \rho \mathbf{C} \mathbf{x}^{(k+1)}. \tag{5.11}$$

It follows by combining (5.9) and (5.11) that

$$\rho \mathbf{C} \mathbf{x}^{(k+1)} = \left(\mathbf{I} + \rho \frac{\mathbf{C}^{\dagger T}}{\rho} \nabla g \frac{\mathbf{C}^{\dagger}}{\rho} \right)^{-1} \Pi_{\text{ran}(\mathbf{C})} \left(\mathbf{z}^{(k)} + \rho \mathbf{d} \right). \tag{5.12}$$

Using Eq. (5.12), the crux of the ellipsoidal bound is summarized in the following Lemma.

Lemma 5.4.1. *For* g μ*-strongly convex and* β*-smooth, and* $\rho \in (0, +\infty)$*, the operator* $\left(\mathbf{I} + \rho \frac{\mathbf{C}^{\dagger T}}{\rho} \nabla g \frac{\mathbf{C}^{\dagger}}{\rho} \right)^{-1} - \hat{\mu} \mathbf{I}$ *is* $\frac{1}{\hat{\beta} - \hat{\mu}}$ *cocoercive over* $\text{ran}\,(\mathbf{C})$*, where*

$$\hat{\beta} = \frac{1}{1 + \frac{\mu}{\rho \sigma^2_{\max}(C)}}, \quad \hat{\mu} = \frac{1}{1 + \frac{\beta}{\rho \sigma^2_{\min \neq 0}(C)}}.$$

The proof of this Lemma, which closely follows the approach adopted in [82, Proposition 2], can be found in Appendix 5.C.

From the definition of cocoercivity and the PDMM updates, we can then form the final Lemma for the ellipsoidal bond of the primal $\mathbf{y}^{(k+1)}$ variables.

Lemma 5.4.2. *The primal updates* $\mathbf{y}^{(k+1)}$ *of PDMM as defined in Algorithm 6 satisfy the inequality*

$$\left\| \prod_{\text{ran}(\mathbf{C})} \left(\mathbf{y}^{(k+1)} - \mathbf{y}^* - (1-\hat{\beta}-\hat{\mu}) \left(\mathbf{z}^{(k)} - \mathbf{z}^* \right) \right) \right\| \leq (\hat{\beta}-\hat{\mu}) \left\| \prod_{\text{ran}(\mathbf{C})} \left(\mathbf{z}^{(k)} - \mathbf{z}^* \right) \right\|.$$

The proof for this lemma can be found in Appendix 5.D, allowing us to form the complete error bound given in (5.6a). This ellipsoidal bound demonstrates how the local updates at each node cause a contraction of the variables within the desired subspace ran $(\mathbf{C})$.

5.4.6. PRINCIPAL ANGLES AND ALTERNATING PROJECTIONS

The second component needed to form our bounding sequence for PDMM is an expression of how the permutation operation $\mathbf{P}$ affects the ellipsoidal bound in (5.6). In particular, $\mathbf{P}$ mixes the subspaces ran $(\mathbf{C})$ and ker $\left(\mathbf{C}^T\right)$ degrading the contraction introduced by the primal updates.

To understand the mixing effect of $\mathbf{P}$, we can note that

$$\mathbf{P} = \frac{\mathbf{I}+\mathbf{P}}{2} - \frac{\mathbf{I}-\mathbf{P}}{2} = \prod_{\ker(\mathbf{I}-\mathbf{P})} - \prod_{\text{ran}(\mathbf{I}-\mathbf{P})} = \mathbf{I} - 2 \prod_{\text{ran}(\mathbf{I}-\mathbf{P})}. \tag{5.13}$$

Thus, the subspace ran$(\mathbf{I}-\mathbf{P})$ is the component affected by the permutation operation. We can then use the notion of principal angles between subspaces to analyze the aforementioned mixing. This notion was first proposed by Jordan [107] and generalizes that of angles between lines in a Euclidean space. In this way, the principal angles between ran$(\mathbf{I}-\mathbf{P})$ and ran $(\mathbf{C})$ can be used to quantify the aforementioned mixing effect. Notably, we can exploit recent results on optimal rate bounds for generalizations of the classic alternating projections algorithm [88],[89] to achieve this point. This observation was previously noted in [99] to construct the geometric convergence proof for PDMM therein. In conjunction with the analysis of the previous Section, in this work we extend upon this connection to provide tighter convergence results.

Consider the two projection operations $\prod_{\text{ran}(\mathbf{C})}$ and $\prod_{\text{ran}(\mathbf{I}-\mathbf{P})}$. It was shown in [89], that there exists an orthonormal matrix $\mathbf{D}$ such that these two projections can be jointly factorized as

$$\prod_{\text{ran}(\mathbf{C})} = \mathbf{D} \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{D}^T, \quad \prod_{\text{ran}(\mathbf{I}-\mathbf{P})} = \mathbf{D} \begin{bmatrix} \cos^2(\boldsymbol{\theta}) & \cos(\boldsymbol{\theta})\sin(\boldsymbol{\theta}) & \mathbf{0} & \mathbf{0} \\ \cos(\boldsymbol{\theta})\sin(\boldsymbol{\theta}) & \sin^2(\boldsymbol{\theta}) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{D}^T,$$

where $\boldsymbol{\theta}$ is a vector of principal angles between ran$(\mathbf{C})$ and ran$(\mathbf{I}-\mathbf{P})$. Similarly, $\cos(\boldsymbol{\theta})$ and $\sin(\boldsymbol{\theta})$ denote diagonal matrices of the cosines and sines of $\boldsymbol{\theta}$ respectively. Using (5.13), the operator $\mathbf{P}$ is therefore given by

$$\mathbf{P} = \mathbf{D} \begin{bmatrix} -\cos(2\boldsymbol{\theta}) & -\sin(2\boldsymbol{\theta}) & \mathbf{0} & \mathbf{0} \\ -\sin(2\boldsymbol{\theta}) & \cos(2\boldsymbol{\theta}) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{D}^T.$$

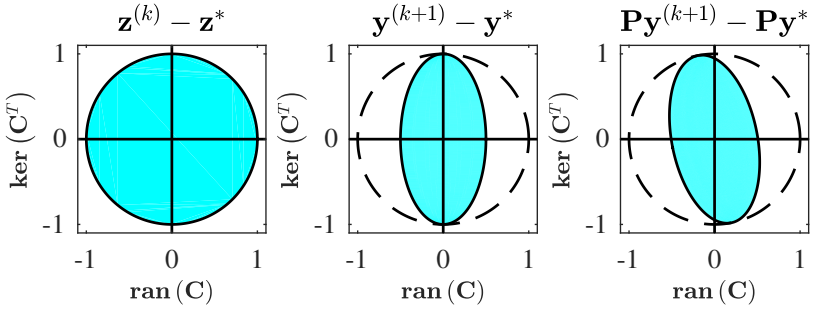


Figure 5.1: Demonstration of the ellipsoidal bound of PDMM updates for the case of $\hat{\mu} = 1 - \hat{\beta} = 0.25$ and $\theta_w = \frac{\pi}{32}$.

The lower right hand identity matrices correspond to those non-contractive components of PDMM and as such do not contribute to the projected auxiliary variables. Similarly, for any $\theta \in \{0, \frac{\pi}{2}\}$, no mixing effect is introduced between ran $(\mathbf{C})$ and ker $\left(\mathbf{C}^T\right)$ and, as we will see in the coming subsections, no degradation of the partial contraction occurs.

5.4.7. TOWARDS A STRONGER CONVERGENCE RATE BOUND FOR PDMM

By combining our analysis of the primal and auxiliary updates defined in the previous sections, we are finally ready to form our new worst-case bound for the projected auxiliary error of PDMM. For a given k iterations, bounding the worst-case convergence can be rephrased as the following non-convex optimization problem:

$$\max_{\mathbf{z}^{(0)}} \quad \left\| \prod_{\text{ran}(\mathbf{PC})+\text{ran}(\mathbf{C})} \left(\mathbf{z}^{(k+1)} - \mathbf{z}^*\right) \right\|^2 \tag{5.14a}$$

$$\text{s.t.} \quad \left\|\mathbf{z}^{(0)} - \mathbf{z}^*\right\|^2 \leq \epsilon_0 \tag{5.14b}$$

$$\left\| \prod_{\text{ran}(\mathbf{C})} \left(\mathbf{y}^{(i+1)} - \mathbf{y}^* - (1-\hat{\beta}-\hat{\mu})\left(\mathbf{z}^{(i)} - \mathbf{z}^*\right)\right) \right\| \leq (\hat{\beta}-\hat{\mu}) \left\| \prod_{\text{ran}(\mathbf{C})} \left(\mathbf{z}^{(i)} - \mathbf{z}^*\right) \right\|, \tag{5.14c}$$

$$\prod_{\ker(\mathbf{C}^T)} \left(\mathbf{y}^{(i+1)} - \mathbf{y}^*\right) = \prod_{\ker(\mathbf{C}^T)} \left(\mathbf{z}^{(i)} - \mathbf{z}^*\right), \tag{5.14d}$$

$$\mathbf{z}^{(i+1)} = \mathbf{P}\mathbf{y}^{(i+1)} \quad \forall i = 0, 1, \cdots, k. \tag{5.14e}$$

Using the results of Section 5.4.2, here we are maximizing the projected auxiliary error (5.14a) under the constraints (5.14c) and (5.14d) imposed by the primal ellipsoidal bounds in (5.6), and the auxiliary update equations of PDMM (5.14e). Additionally, (5.14b) bounds the initial error.

To simplify (5.14), reconsider the joint decomposition of ran$(\mathbf{C})$ and ran$(\mathbf{I}-\mathbf{P})$ defined in Section 5.4.6. We can use this decomposition and the resulting link between $\boldsymbol{\theta}$ and the matrix $\mathbf{P}$ to perform a change of variables that simplifies the matrix multiplication in

(5.14e), allowing us to compute the maximum of (5.14). Specifically, by defining $\hat{\mathbf{z}}^{(i)} = \mathbf{D}^T\mathbf{z}^{(i)}$ and $\hat{\mathbf{y}}^{(i)} = \mathbf{D}^T\mathbf{y}^{(i)}$, (5.14e) is equivalent to

$$\hat{\mathbf{z}}^{(i+1)} = \begin{bmatrix} -\cos(2\boldsymbol{\theta}) & -\sin(2\boldsymbol{\theta}) & \mathbf{0} & \mathbf{0} \\ -\sin(2\boldsymbol{\theta}) & \cos(2\boldsymbol{\theta}) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \hat{\mathbf{y}}^{(i+1)}. \tag{5.15}$$

Importantly, as $\cos(2\boldsymbol{\theta})$ and $\sin(2\boldsymbol{\theta})$ are diagonal matrices, the submatrix $\begin{bmatrix} -\cos(2\boldsymbol{\theta}) & -\sin(2\boldsymbol{\theta}) \\ -\sin(2\boldsymbol{\theta}) & \cos(2\boldsymbol{\theta}) \end{bmatrix}$ is symmetrically permutable to a two by two block diagonal form.

As (5.14) is the maximization of a convex quadratically constrained quadratic problem, its worst-case solution will lie on the boundary of its constraint set. Furthermore, due to the block structure in (5.15), the worst case solution must also correspond to a particular two dimensional subspace parameterized by a worst-case principal angle θ_w which is an element of $\boldsymbol{\theta}$. We can equivalently compute the maximum of (5.14) via the solution of the two dimensional optimization problem

$$\begin{aligned} \max_{z_R^{(0)}, z_K^{(0)}} \quad & |z_R^{(k+1)} - z_R^*|^2 + |z_K^{(k+1)} - z_K^*|^2 \\ \text{s.t.} \quad & |z_R^{(0)} - z_R^*|^2 + |z_K^{(0)} - z_K^*|^2 \le \epsilon_0 \\ & |y^{(i+1)} - (1-\hat{\beta}-\hat{\mu}) z_R^{(i)}| \le (\hat{\beta}-\hat{\mu})|z_R^{(i)}| \\ & \begin{bmatrix} z_R^{(i+1)} \\ z_K^{(i+1)} \end{bmatrix} = \begin{bmatrix} -\cos(2\theta_w) & -\sin(2\theta_w) \\ -\sin(2\theta_w) & \cos(2\theta_w) \end{bmatrix} \begin{bmatrix} y^{(i+1)} \\ z_K(i) \end{bmatrix} \quad \forall i = 0, 1, \cdots, k, \end{aligned} \tag{5.16}$$

where the scalars $y^{(i)}, z_R^{(i)}$ represent those components of $\mathbf{y}^{(i)}$ and $\mathbf{z}^{(i)}$ that coincide with $\text{ran}(\mathbf{C})$, while $z_K^{(i)}$ denotes the component of $\mathbf{z}^{(i)}$ in $\ker(\mathbf{C}^\mathbf{T})$. Our ability to omit the projection in the objective of (5.16) stems from the fact that an optimizer of (5.14) must correspond to an initial vector $\mathbf{z}^{(0)}$ contained within $\text{ran}(\mathbf{PC}) + \text{ran}(\mathbf{C})$ due to the use of the projected auxiliary error introduced in Section 5.4.2. Hence, $\exists \mathbf{z}^* \mid \mathbf{z}^{(k+1)} - \mathbf{z}^* \in \text{ran}(\mathbf{PC}) + \text{ran}(\mathbf{C})$. To further simplify things, we have also made explicit use of (5.14d) to remove the need for a separate variable to model the component of $\mathbf{y}$ in $\ker(\mathbf{C}^\mathbf{T})$. The proposed reformulation is also attractive as it explicitly partitions the auxiliary variables into those components which influence the primal $\mathbf{x}$ variables and those which do not.

For the worst-case iterates, the ellipsoidal constraints in (5.16) hold with equality. By defining the scalar δ as

$$\delta = \begin{cases} 1-2\hat{\beta} & \text{if } 2\hat{\beta} - 1 > 1 - 2\hat{\mu} \\ 1-2\hat{\mu} & \text{otherwise} \end{cases}, \tag{5.17}$$

it follows that $y^{(i+1)} = \delta z_R^{(i)} \ \forall i = 0, 1, \cdots, k$.

Substituting δ into (5.16), we find that for a given $i = 1, \cdots, k$ the worst-case updates are given by

$$\begin{bmatrix} z_R^{(i+1)} \\ z_K^{(i+1)} \end{bmatrix} = \boldsymbol{\Phi} \begin{bmatrix} z_R^{(i)} \\ z_K^{(i)} \end{bmatrix}, \tag{5.18}$$

where the matrix $\mathbf{\Phi}$ is defined as

$$\mathbf{\Phi} = \begin{bmatrix} -\cos(2\theta_w) & -\sin(2\theta_w) \\ -\sin(2\theta_w) & \cos(2\theta_w) \end{bmatrix} \begin{bmatrix} \delta & 0 \\ 0 & 1 \end{bmatrix}. \tag{5.19}$$

As our analysis thus far would suggest, the matrix $\mathbf{\Phi}$ is comprised of two factors; a contraction factor (δ) dependent on the local functions at each node, and a mixing effect (θ_w) due to the edge based constraint functions. In the case of the two dimensional problem considered in (5.16), the mixing due to θ_w results in a rotation of the error ellipsoid at each iteration, as demonstrated in Figure 5.1. In the following we demonstrate how this rotation influences the convergence rate of the projected auxiliary variables which in turn allows us to define the worst case principal angle θ_w.

5.4.8. WORST-CASE CONVERGENCE BOUND AND ITS LIMITING RATE

As the worst-case auxiliary updates given in (5.18) all hold simultaneously, the solution of (5.14) satisfies the inequality

$$\left\| \Pi_{\text{ran}(\mathbf{PC})+\text{ran}(\mathbf{C})} \left(\mathbf{z}^{(k+1)} - \mathbf{z}^* \right) \right\|^2 = \sigma^2_{\max}\left(\mathbf{\Phi}^k\right) \epsilon_0, \tag{5.20}$$

where $\sigma^2_{\max}(\bullet)$ denotes the maximum squared singular value of a matrix. Note that as the bound given in (5.20) is based on the largest singular value, the worst-case rates for different k need not correspond to the same initial vector $\mathbf{z}^{(0)}$ or even the same worst-case function. Like the primal variable error ellipsoids demonstrated in Figure 5.1, (5.20) can also be visualized via a sequence of rotated error ellipsoids. An example of this point is demonstrated in Figure 5.2 for a particular instance of PDMM. In this way, (5.20) is an *iteration specific* bound.

The worst case principal angle θ_w corresponds to the element of $\boldsymbol{\theta}$ which maximizes the largest singular value of $\mathbf{\Phi}^k$. As $\theta_w \in [0, \frac{\pi}{2}]$ by definition, one of two cases holds: either $\theta_w \in \{0, \frac{\pi}{2}\}$ and lies on the boundary of the set, or $\theta_w \in (0, \frac{\pi}{2})$ and is contained within the set. If we assume the prior, from (5.19) we can note that $\mathbf{\Phi} = \pm \begin{bmatrix} \delta & 0 \\ 0 & 1 \end{bmatrix}$, and thus no mixing occurs between the subspaces $\text{ran}(\mathbf{C})$ and $\ker(\mathbf{C})$. This lack of mixing enables us to immediately note that

$$\left\| \Pi_{\text{ran}(\mathbf{C})} \left(\mathbf{z}^{(k+1)} - \mathbf{z}^* \right) \right\|^2 \leq |\delta|^{2k} \epsilon_0, \tag{5.21}$$

which follows from the structure of $\mathbf{\Phi}$ and the definition of z_R used in (5.16). We are therefore able to directly bound the error in the auxiliary variables which influence the primal updates.

Using (5.21), we can demonstrate that any $\theta_w \in \left(0, \frac{\pi}{2}\right)$ must result in slower convergence than a $\theta_w \in \{0, \frac{\pi}{2}\}$ by considering the *limiting rate* of PDMM given by

$$\gamma_w = \lim_{k\to\infty} \sqrt[k]{\sigma_{\max}\left(\mathbf{\Phi}^k\right)} \leq \sqrt[k]{\sigma_{\max}\left(\mathbf{\Phi}^k\right)} \quad \forall k \in \mathbb{N}. \tag{5.22}$$

Here, γ_w describes the average contraction per iteration as the number of iterations k tends to infinity, hence the use of the term limiting rate. By definition, this directly corresponds to the spectral radius of $\mathbf{\Phi}$, its largest absolute eigenvalue.

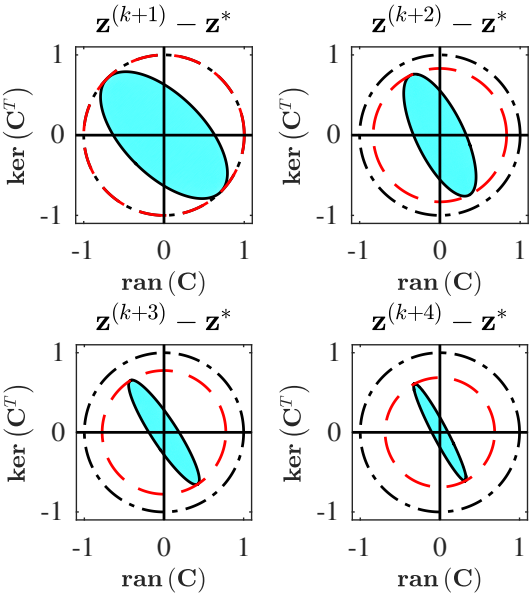


Figure 5.2: Demonstration of the contractive nature of PDMM over multiple iterations for the case of $\hat{\mu} = \hat{\beta} = 0.25$ and $\theta_w = \frac{\pi}{8}$. The red dotted line demonstrates the worst-case convergence bound.

We can compute γ_w by using the fact that $\mathbf{\Phi} \in \mathbb{R}^{2\times 2}$ and thus that its eigenvalues $\lambda(\mathbf{\Phi})$ can be computed analytically from its trace and determinant given by

$$\mathrm{tr}(\mathbf{\Phi}) = (1-\delta)\cos(2\theta_w), \ \det(\mathbf{\Phi}) = -\delta.$$

It follows that $\lambda(\mathbf{\Phi})$ can be computed as

$$\lambda(\mathbf{\Phi}) = \frac{(1-\delta)}{2}\cos(2\theta_w) \pm \sqrt{\frac{(1-\delta)^2}{4}\cos^2(2\theta_w) + \delta}.$$

The limiting rate of PDMM is therefore given by

$$\gamma_w = \max\{|\lambda(\mathbf{\Phi})|\}.$$

As $\delta \in (-1,1)$, we can form an analytic expression for γ_w in terms of $|\delta|$ and $|\cos(2\theta_w)|$. By defining $\alpha_1 = \frac{(1-|\delta|)}{4}^2 |\cos(2\theta_w)|^2, \alpha_2 = \frac{(1+|\delta|)}{4}^2 |\cos(2\theta_w)|^2$ it follows that

$$\gamma_w = \begin{cases} \frac{1-|\delta|}{2}|\cos(2\theta_w)| + \sqrt{\alpha_1 + |\delta|} & \delta \in [0,1) \\ \frac{1+|\delta|}{2}|\cos(2\theta_w)| + \sqrt{\alpha_2 - |\delta|} & \delta \in [-\alpha_2, 0) \\ \sqrt{|\delta|} & \delta \in (-1, -\alpha_2) \end{cases}, \tag{5.23}$$

where an increase in either $|\delta|$ or $|\cos(2\theta_w)|$, results in an increase in γ_w. For all such δ, we therefore find that $\gamma_w \geq \sqrt{|\delta|} \geq |\delta|$ where the final equality follows as $\delta \in (-1,1)$.

By combining (5.22) and (5.23), we find that $\delta^{2k} \leq \gamma_w^{2k} \leq \sigma_{\max}^2\left(\mathbf{\Phi}^k\right)$ and thus that $\theta_w \in (0, \frac{\pi}{2})$ must hold, assuming such a principal angle exists. Furthermore, as γ_w increases with both $|\delta|$ and $|\cos(2\theta_w)|$, θ_w should be chosen as close to but not equal not to 0 or $\frac{\pi}{2}$ to achieve the worst case rate.

Remark 1. *In the case that all principal angles are contained within the set* $\{0, \frac{\pi}{2}\}$*, it implies that the matrix* $\underset{\text{ran}(\mathbf{I}-\mathbf{P})}{\Pi}$ *and* $\underset{\text{ran}(\mathbf{C})}{\Pi}$ *are jointly diagonalizable. Due to the node block structure of* $\mathbf{C}$ *and the edge structure of* $\mathbf{P}$*, this can only occur in the case that the constraint functions impose no form of coupling between the local variables at each node, i.e. each node operates entirely independently. As such a problem is not a true example of distributed optimization, we can say that* $\theta_w \in (0, \frac{\pi}{2})$ *for all distributed optimization problems.*

5.4.9. OPTIMAL STEP SIZE CHOICE FOR A GIVEN NETWORK

In addition to understanding the convergence characteristics of PDMM, we can also demonstrate a method of step size selection to optimize the worst-case convergence bound. This is achieved by noting that, while the choice of the step size ρ influences the configuration of $\hat{\mu}$ and $\hat{\beta}$ (see Lemma 5.4.1) it does not affect θ_w which only depends on the matrices $\mathbf{C}$ and $\mathbf{P}$. We can therefore configure the step size to optimize $|\delta|$ and thus the worst case bound. Note that following choice was initially provided in [82] but is included here for completeness.

The optimal choice of ρ should minimize the magnitude of local contraction factor δ as defined in (5.17). Specifically, as $\hat{\beta} \geq \hat{\mu}$, such a step size corresponds to minimizer of

$$|\delta_{\text{opt}}| = \min_{\rho>0} \max\left\{ \frac{2}{1+\frac{\mu}{\rho\sigma_{\max}^2(C)}} - 1, 1 - \frac{2}{1+\frac{\beta}{\rho\sigma_{\min\neq 0}^2(C)}} \right\}.$$

With some basic manipulation, it also follows that

$$|\delta_{\text{opt}}| = \min_{\rho>0} \max\left\{ \frac{\rho\frac{\sigma_{\max}^2(C)}{\mu} - 1}{\rho\frac{\sigma_{\max}^2(C)}{\mu} + 1}, \frac{1 - \rho\frac{\sigma_{\min\neq 0}^2(C)}{\beta}}{\rho\frac{\sigma_{\min\neq 0}^2(C)}{\beta} + 1} \right\}. \tag{5.24}$$

By inspection, the optimal choice of ρ occurs when the two terms in (5.24) are equal, i.e., that

$$\frac{\rho_{\text{opt}}\frac{\sigma_{\max}^2(C)}{\mu} - 1}{\rho_{\text{opt}}\frac{\sigma_{\max}^2(C)}{\mu} + 1} = \frac{1 - \rho_{\text{opt}}\frac{\sigma_{\min\neq 0}^2(C)}{\beta}}{\rho_{\text{opt}}\frac{\sigma_{\min\neq 0}^2(C)}{\beta} + 1}.$$

Rearranging this equality and cancelling common terms, it follows that the optimal step size is given by

$$\rho_{\text{opt}} = \frac{\sqrt{\mu\beta}}{\sigma_{\max}(C)\,\sigma_{\min\neq 0}(C)}. \tag{5.25}$$

Substituting this into the definition of δ in (5.17), we find that

$$|\delta_{\text{opt}}| = \frac{\sigma_{\max}(C)\sqrt{\beta} - \sigma_{\min\neq 0}(C)\sqrt{\mu}}{\sigma_{\max}(C)\sqrt{\beta} + \sigma_{\min\neq 0}(C)\sqrt{\mu}}. \tag{5.26}$$

While this particular ρ optimizes the convergence rate bound, it need not be optimal for a particular problem instance.

5.5. ADDITIONAL ANALYSIS AND RESULTS

In this section we outline a pair of additional results which stem from the newly introduced convergence bound. Specifically we link our new convergence bound with that demonstrated in [99] as well as highlighting a problem instance which converges at the limiting rate γ_w.

5.5.1. THE CONNECTION WITH THE GEOMETRIC BOUND OF PDMM

Using the iteration specific convergence bound introduced above we can re-derive the geometric rate introduced in [99] for the same functional class. We can draw this connection by considering the specific case when $k=2$. Specifically, the $k=2$-iteration specific bound is parameterized by the matrix

$$\boldsymbol{\Phi}^2 = \begin{bmatrix} \delta^2+\delta(1-\delta)\sin^2(2\theta_w) & (\delta-1)\cos(2\theta_w)\sin(2\theta_w) \\ \delta(\delta-1)\cos(2\theta_w)\sin(2\theta_w) & \delta+(1-\delta)\cos^2(2\theta_w) \end{bmatrix}.$$

As this is a two by two matrix, we can determine its singular values from the eigenvalues of $\left(\boldsymbol{\Phi}^2\right)^T\boldsymbol{\Phi}^2$ by using the same analysis adopted in Section 5.4.8 i.e. from its trace and determinant which are respectively by

$$\mathrm{tr}\left(\boldsymbol{\Phi}^2\right) = 2\delta^2 + \left(1-\delta^2\right)^2\cos^2(2\theta_w),\ \det\left(\boldsymbol{\Phi}^2\right) = \delta^4.$$

It follows that the squared singular values of $\boldsymbol{\Phi}^2$ are given by

$$\sigma^2\left(\boldsymbol{\Phi}^2\right) = \delta^2 + \frac{\left(1-\delta^2\right)^2\cos^2(2\theta_w)}{2} \pm \left(1-\delta^2\right)|\cos(2\theta_w)|\sqrt{\frac{\left(1-\delta^2\right)^2\cos^2(2\theta_w)}{4}+\delta^2} \quad (5.27)$$

Comparing the result in (5.27) with that in [99, Lemma V.2], we can observe that the convergence rate therein is identical if we apply the mapping $|\cos(2\theta_w)| = \cos(\theta_F)$. It follows that the convergence rate bound of [99] is only tight for a single two-iterate update. This leads to a reduction in the worst-case rate, particularly for those with favorable θ_w and ω parameters.

Figure 5.3 highlights this difference in convergence bound by comparing the geometric rates of [99] and that provided in (5.20). We can clearly note the stronger rate provided by the iteration specific bound with equality between the two bounds for the instance that $k=2$. Figure 5.3 also demonstrates the limiting convergence rate γ_w given in (5.23) with the iteration specific bound tending to this rate as k increases.

5.5.2. A PROBLEM INSTANCE THAT ATTAINS THE WORST-CASE RATE

In addition to drawing a connection with the existing geometric convergence bound of PDMM in the literature, we can also show that the limiting rate of PDMM is in fact tight by demonstrating a particular problem instance which converges at this rate. To do so, consider a d-regular network where every node has d edges. For each node i we define

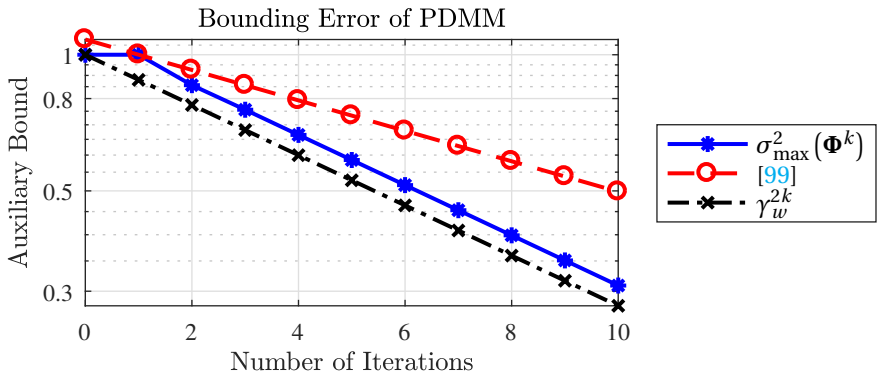


Figure 5.3: A comparison of the convergence rates of the proposed tight bound and the existing geometric convergence bound of PDMM in [99] for the case of $\omega = 0.55$ and $\theta_w = \frac{\pi}{16}$.

the local objective function $f_i = \frac{1}{2}\mathbf{x}_i^T \begin{bmatrix} \mu & 0 \\ 0 & \beta \end{bmatrix} \mathbf{x}_i$ with $0 \leq \mu \leq \beta$. Importantly, this ensures that all local objectives are μ-strongly convex and β-smooth. Using this configuration, consider solving the following problem with PDMM.

$$\min_{\mathbf{x}} \sum_{i \in V} \frac{1}{2}\mathbf{x}_i^T \begin{bmatrix} \mu & 0 \\ 0 & \beta \end{bmatrix} \mathbf{x}_i \quad \text{s.t}\, \mathbf{x}_i = \mathbf{x}_j \quad \forall (i,j) \in E., \tag{5.28}$$

where the constraints impose consensus between the local variables. In the following, we demonstrate how the rate of PDMM is given by γ_w for this particular problem.

To determine the convergence rate of PDMM in solving (5.28) we can consider the update equations given in Algorithm 6. Specifically, due to the quadratic nature of the objective functions, the update equations for PDMM are linear. We can therefore express PDMM via the single update equation

$$\mathbf{z}^{(k+1)} = \mathbf{P}\left(\mathbf{I} - 2\rho\mathbf{C}\left(\mathbf{Q} + \rho\mathbf{C}^T\mathbf{C}\right)^{-1}\mathbf{C}^T\right)\mathbf{z}^{(k)},$$

where $\mathbf{Q} = \mathbf{I}_N \otimes \begin{bmatrix} \mu & 0 \\ 0 & \beta \end{bmatrix}$ and $\otimes$ denotes the Kronecker product. The linear nature of this equation allows us to directly determine the convergence rate of PDMM. Specifically, as the effect of the permutation matrix $\mathbf{P}$ has already been addressed in Section 5.4.6, it follows that all we need to verify to prove the tightness of (5.23) is to show that $\rho\mathbf{C}\left(\mathbf{Q} + \rho\mathbf{C}^T\mathbf{C}\right)^{-1}\mathbf{C}^T$ is $\hat{\beta}$ Lipschitz continuous and $\hat{\mu}$ strongly monotone over ran $(\mathbf{C})$.

Denoting the diagonal *degree matrix* of graph G by $\mathbf{D}_G$, where each diagonal element d_i denotes the degree of node i, i.e., how many edges include i as a vertex, it follows that

$$\mathbf{C}^T\mathbf{C} = \mathbf{D}_G \otimes \mathbf{I}_M, \tag{5.29}$$

where we have used the definition of $\mathbf{C}$ in (5.3) noting that $\mathbf{A}_{i|j} = \pm\mathbf{I}$ for consensus problems. Furthermore, in the case of the networks considered in (5.28), $\mathbf{D}_G = d\mathbf{I}$. It follows that for a connected network, $\mathbf{C}$ has full column rank. Thus the eigenvectors of

$\mathbf{I}-2\rho\mathbf{C}\left(\mathbf{Q}+\rho\mathbf{C}^T\mathbf{C}\right)^{-1}\mathbf{C}^T$ whose eigenvalues are equal to 1 lie outside of ran$(\mathbf{C})$. Therefore using [108, Theorem 1.3.20], $\mathbf{I}-2\rho\mathbf{C}\left(\mathbf{Q}+\rho\mathbf{C}^T\mathbf{C}\right)^{-1}\mathbf{C}^T$ has the same non-unity eigenvalues as the smaller matrix

$$\mathbf{I}-2\rho\mathbf{C}^T\mathbf{C}\left(\mathbf{Q}+\rho\mathbf{C}^T\mathbf{C}\right)^{-1}=\mathbf{I}_N\otimes\begin{bmatrix}1-\frac{2}{1+\frac{\mu}{\rho d}} & 0\\ 0 & 1-\frac{2}{1+\frac{\beta}{\rho d}}\end{bmatrix}. \tag{5.30}$$

Recalling from Lemma 5.4.1 that

$$\hat{\beta}=\frac{1}{1+\frac{\mu}{\rho\sigma^2_{\max}(C)}},\ \hat{\mu}=\frac{1}{1+\frac{\beta}{\rho\sigma^2_{\min\neq 0}(C)}},$$

in combination with the structure of (5.30), we can conclude that $\rho\mathbf{C}\left(\mathbf{Q}+\rho\mathbf{C}^T\mathbf{C}\right)^{-1}\mathbf{C}^T$ is $\hat{\beta}$-Lipschitz and $\hat{\mu}$-strongly monotone over ran$(\mathbf{C})$ as desired. It follows that the convergence rate of PDMM is exactly given by (5.23) in this instance.

5

5.6. THE EFFECT OF NETWORK TOPOLOGY ON DISTRIBUTED CONSENSUS

Given our analysis of the convergence rate of PDMM, we are finally able to quantify how network topology influences algorithm performance. Specifically, we will show how the convergence rate of consensus problems depends on both the minimum and maximum degree of any node in the network, and the spectrum of the random walk matrix of the underlying graph. Using this connection, we analyze the characteristics of a number of different deterministic networks allowing us to compare their worst case convergence. In particular, this allows us to demonstrate a problem instance where PDMM converges in a finite number of iterations.

5.6.1. THE INTERPLAY BETWEEN CONSENSUS AND TOPOLOGY

Consider a consensus problem of the form

$$\min_{\mathbf{x}_i\,\forall\, i\in V}\sum_{i\in V}f_i(\mathbf{x}_i)\quad \text{s.t}\,\mathbf{x}_i=\mathbf{x}_j\quad\forall\,(i,j)\in E, \tag{5.31}$$

where $M_i=M\ \forall i\in V$. While such problems are more restrictive than those able to be solved via PDMM, our motivation for this choice stems from the link that emerges between algorithm convergence and spectral graph theory in this case. Specifically, we show how the spectrum of the random walk matrix of G, which we shall define shortly, determines $\cos(2\theta_w)$.

For the considered class of functions, we have previously seen that the limiting convergence rate bound in (5.20) is parameterized by two key terms: the partial contraction factor δ which is dependent of μ,β,ρ and the singular values of $\mathbf{C}$, and the worst case principle angle θ_w which is dependent on the matrices $\mathbf{C}$ and $\mathbf{P}$. As the edge set E of the underlying graph determines the form of the constraints in (5.31) it follows that both $\mathbf{C}$ and $\mathbf{P}$ and thus δ and θ_w are dependent on the underlying topology. These effects are quantified below.

THE EFFECT ON THE LOCAL CONTRACTION PARAMETER δ

Considering first the local contraction factor δ, for a given μ, β, ρ and the considered class of consensus problems, dependence on topology follows from (5.29) where we can note that $\sigma_{\min\neq 0}(\mathbf{C}) = \sqrt{d_{\min}}$, $\sigma_{\max}(\mathbf{C}) = \sqrt{d_{\max}}$ are the squareroots of the minimum and maximum degree of any node in the network respectively. Therefore, if we select the stepsize ρ to optimize $|\delta|$, as per Section 5.4.9, it follows from (5.26) that

$$|\delta| = \frac{\sqrt{d_{\max}\beta} - \sqrt{d_{\min}\mu}}{\sqrt{d_{\max}\beta} + \sqrt{d_{\min}\mu}}. \tag{5.32}$$

For a fixed β and μ, an increase in the degree spread $d_{\max} - d_{\min}$ increases $|\delta|$ and thus reduces convergence rate. In this way, d-regular graphs, those where all nodes have the same degree, potentially offer faster convergence bounds.

THE EFFECT ON THE SUBSPACE MIXING PARAMETER $\cos(2\theta_w)$

We can also infer the effect of topology on $\cos(2\theta_w)$ by using the definition of the cosines of the principal angles between $\text{ran}(\mathbf{C})$ and $\text{ran}(\mathbf{I}-\mathbf{P})$. Specifically, from [89, Definition 3.1], the diagonal elements of the matrix $\cos(\boldsymbol{\theta})$ are given by the singular values of the matrix product

$$\mathbf{S} = \prod_{\text{ran}(\mathbf{I}-\mathbf{P})} \prod_{\text{ran}(\mathbf{C})}.$$

We can compute these singular values and thus $\cos(\boldsymbol{\theta})$ as the square roots of the eigenvalues of the matrix

$$\begin{aligned}\mathbf{S}^T\mathbf{S} &= \prod_{\text{ran}(\mathbf{C})} \prod_{\text{ran}(\mathbf{I}-\mathbf{P})} \prod_{\text{ran}(\mathbf{C})} = \frac{1}{2}\left(\prod_{\text{ran}(\mathbf{C})} - \prod_{\text{ran}(\mathbf{C})} \mathbf{P} \prod_{\text{ran}(\mathbf{C})}\right) \\ &= \frac{1}{2}\mathbf{C}\left(\left(\mathbf{C}^T\mathbf{C}\right)^\dagger - \left(\mathbf{C}^T\mathbf{C}\right)^\dagger \mathbf{C}^T\mathbf{P}\mathbf{C}\left(\mathbf{C}^T\mathbf{C}\right)^\dagger\right)\mathbf{C}^T,\end{aligned}$$

where we have used the definition of orthonormal projections.

To analyze $\mathbf{S}^T\mathbf{S}$ we can again use [108, Theorem 1.3.22], such that the non-zero eigenvalues of $\mathbf{S}^T\mathbf{S}$ can equivalently be found via the spectrum of the smaller matrix

$$\begin{aligned}\mathscr{S} &= \frac{1}{2}\left(\left(\mathbf{C}^T\mathbf{C}\right)^\dagger - \left(\mathbf{C}^T\mathbf{C}\right)^\dagger \mathbf{C}^T\mathbf{P}\mathbf{C}\left(\mathbf{C}^T\mathbf{C}\right)^\dagger\right)\mathbf{C}^T\mathbf{C} \\ &= \frac{1}{2}\left(\mathbf{I} - \left(\mathbf{D}_G^{-1} \otimes \mathbf{I}_M\right)\mathbf{C}^T\mathbf{P}\mathbf{C}\right),\end{aligned}$$

where we have also used (5.29) to simplify notation.

To better interpret the structure of $\mathscr{S}$ we require the additional notion of the *adjacency matrix* of a graph. Specifically, for a given G, this matrix is denoted by $\mathbf{A}_G \in \mathbb{R}^{N\times N}$ where $[\mathbf{A}_G]_{i,j} = 1$ if $(i,j) \in E$ and is zero otherwise. Using this definition, it follows that $\mathbf{C}^T\mathbf{P}\mathbf{C} = -\mathbf{A}_G \otimes \mathbf{I}_M$ where the negative sign stems from the fact that $\forall (i,j) \in E$, $\mathbf{A}_{i|j} = -\mathbf{A}_{j|i} \in \{\pm\mathbf{I}\}$. Thus, using the mixed product property of Kronecker products, it follows that

$$\mathscr{S} = \frac{1}{2}\left(\mathbf{I} + \mathbf{D}_G^{-1}\mathbf{A}_G\right) \otimes \mathbf{I}_M.$$

We can further determine a matrix whose spectrum corresponds to $\cos(2\boldsymbol{\theta})$ via simple trigonometric manipulation. Specifically, as $\cos(2\theta) = 2\cos^2(\theta) - 1$ and $\lambda(\mathscr{S}) = \cos^2(\boldsymbol{\theta})$, the eigenvalues of the right stochastic matrix

$$\hat{\mathbf{A}}_G \otimes \mathbf{I}_M = 2\mathscr{S} - \mathbf{I} = \left(\mathbf{D}_G^{-1}\mathbf{A}_G\right) \otimes \mathbf{I}_M,$$

are exactly equal to $\cos(2\boldsymbol{\theta})$. The matrix $\hat{\mathbf{A}}_G$ corresponds to the *random walk matrix* of a graph.

The above link between the random walk matrix and the cosines of principal angles highlights a number of interesting properties. Firstly, applying the Gershgorin circle theorem, the eigenvalues of $\hat{\mathbf{A}}_G$ are contained within $[-1,1]$ which also logically follows from the bounded nature of the cosine function. Furthermore, those eigenvalues with magnitude 1 correspond to $\theta \in \{0, \frac{\pi}{2}\}$ which, for nontrivial distributed problems, do not correspond to the θ_w. It follows that $\cos(2\theta_w)$ is given by the largest non-unity absolute eigenvalue of $\hat{\mathbf{A}}_G$, i.e., by $\max\{|\lambda(\hat{\mathbf{A}}_G)| < 1\}$ where $\lambda(\hat{\mathbf{A}}_G)$ is the spectrum of $\hat{\mathbf{A}}_G$.

That the convergence of PDMM is parameterized by $\max\{|\lambda(\hat{\mathbf{A}}_G)| < 1\}$ provides a direct link with spectral graph theory. In particular, as $\hat{\mathbf{A}}_G$ is right stochastic, it describes the transition probability between nodes during a random walk on the underlying network, hence the name. The overall mixing rate of this random walk is again determined by the largest non-unity absolute eigenvalue of $\hat{\mathbf{A}}_G$ as it describes the slowest converging mode. The convergence rate of PDMM is therefore parameterized by this mixing rate and thus networks which exhibit faster random walks also have the potential to exhibit faster convergence as well.

5.6.2. CONVERGENCE OF DETERMINISTIC NETWORK TOPOLOGIES

Having linked network topology with convergence rate, we now demonstrate the effect of a range of deterministic network structures on the worst case bound given in (5.23). Specifically, for each considered network we select ρ as per (5.25) to optimize γ_w. With this choice, in Table 5.1 we demonstrate analytic expressions for $|\delta|$ and $|\cos(2\theta_w)$ for a range of networks.

Many of the eigenvalue results in Table 5.1 leverage existing results in spectral graph theory, namely that of Chung in [109], by exploiting the link between $\lambda\left(\hat{\mathbf{A}}_G\right)$ and the spectrum of normalized graph Laplacians. Specifically, the normalized Laplacian of a graph G is given by

$$\hat{\mathscr{L}} = \mathbf{I} - \mathbf{D}_G^{\frac{-1}{2}} \mathbf{A}_G \mathbf{D}_G^{\frac{-1}{2}}.$$

Using matrix similarity and the definition of $\hat{\mathbf{A}}_G$, it follows that $\lambda(\hat{\mathbf{A}}_G) = \lambda(\mathbf{D}_G^{-1}\mathbf{A}_G) = \lambda(\mathbf{D}_G^{\frac{-1}{2}} \mathbf{A}_G \mathbf{D}_G^{\frac{-1}{2}}) = \lambda(\mathbf{I} - \hat{\mathscr{L}})$ which provides a direct method of computing the eigenvalues of $\hat{\mathbf{A}}_G$ from the spectrum of $\hat{\mathscr{L}}$. Unlike [109], our results also highlight the interplay between the local functional characteristics and network topology in defining the convergence rate bound for PDMM. In the case of the K-hop lattice and $2D$ periodic grid graphs, these spectra are derived below.

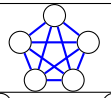
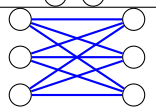
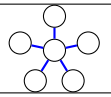
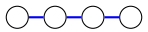
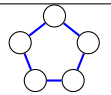
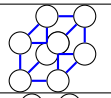
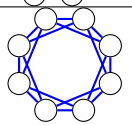
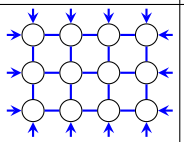
Graph Type:	Example:	$\lvert\delta\rvert$ (5.32)	$\lambda\left(\hat{\mathbf{A}}_G\right)$	$\lvert\cos(2\theta_w)\rvert$
Fully Connected [109, Example 1.1]		$\frac{\sqrt{(N-1)\beta}-\sqrt{(N-1)\mu}}{\sqrt{(N-1)\beta}+\sqrt{(N-1)\mu}}$	$\left\{1, \frac{-1}{N-1}\right\}$	$\frac{1}{N-1}$
Complete Bipartite [109, Example 1.2]. M denotes the number of nodes in the larger partition		$\frac{\sqrt{M\beta}-\sqrt{(N-M)\mu}}{\sqrt{M\beta}+\sqrt{(N-M)\mu}}$	{-1,0,1}	0
Star [109, Example 1.3]		$\frac{\sqrt{(N-1)\beta}-\sqrt{\mu}}{\sqrt{(N-1)\beta}+\sqrt{\mu}}$	$\{-1,0,1\}$	0
Path/Chain [109, Example 1.4]		$\frac{\sqrt{2\beta}-\sqrt{\mu}}{\sqrt{2\beta}+\sqrt{\mu}}$	$\left\{\cos\left(\frac{i\pi}{N-1}\right) \mid i \in \{0,\cdots,N-1\}\right\}$	$\cos\left(\frac{\pi}{N-1}\right)$
Ring [109, Example 1.5]		$\frac{\sqrt{2\beta}-\sqrt{2\mu}}{\sqrt{2\beta}+\sqrt{2\mu}}$	$\left\{\cos\left(\frac{2i\pi}{N}\right) \mid i \in \{0,\cdots,N-1\}\right\}$	$\cos\left(\frac{2\pi}{N}\right)$
K-Cube [109, Example 1.6]		$\frac{\sqrt{K\beta}-\sqrt{K\mu}}{\sqrt{K\beta}+\sqrt{K\mu}}$	$\left\{\frac{K-2i}{K} \mid i \in \{0,\cdots,K\}\right\}$	$\frac{K-2}{K}$
K-Hop Lattice. For $K=1$ the network forms a ring graph.		$\frac{\sqrt{2K\beta}-\sqrt{2K\mu}}{\sqrt{2K\beta}+\sqrt{2K\mu}}$	$\left\{\frac{1}{K}\sum_{j=1}^{K}\cos\left(\frac{2ij\pi}{N}\right) \mid i \in \{0,\cdots,N-1\}\right\}$	$\max\left\{\lvert\lambda\left(\hat{\mathbf{A}}_G\right)\rvert < 1\right\}$
$m \times n$ Grid with Periodic Boundary Condition with $m \geq n$		$\frac{\sqrt{4\beta}-\sqrt{4\mu}}{\sqrt{4\beta}+\sqrt{4\mu}}$	$\left\{\frac{1}{2}\left(\cos\left(\frac{2i\pi}{n}\right)+\cos\left(\frac{2j\pi}{m}\right)\right) \mid i \in \{0,\cdots,n-1\}, j \in \{0,\cdots,m-1\}\right\}$	$\frac{1}{2}\left(\cos\left(\frac{2\pi}{n}\right)+\cos\left(\frac{2\pi}{m}\right)\right)$

Table 5.1: Analytic expressions for the worst-case convergence rate for various network types.

K-HOP LATTICE

The spectrum $\lambda(\hat{\mathbf{A}}_G)$ of a K-hop lattice graph can be derived from the fact that $\hat{\mathbf{A}}_G$ is circulant. Specifically, denoting the first column of $\hat{\mathbf{A}}_G$ by the vector $\mathbf{a} = \frac{1}{2K}[0, \mathbf{1}_K^T, \mathbf{0}_{N-2K-1}^T, \mathbf{1}_K^T]^T$,

$$\lambda\left(\hat{\mathbf{A}}_G\right) = \sum_{j=0}^{N-1}\left(a_j \exp\left(\frac{2ij\pi\sqrt{-1}}{N}\right)\right) \ \forall i \in \{0,\cdots,N-1\}.$$

Using the definition of $\mathbf{a}$ and Euler's rule, it follows that

$$\lambda\left(\hat{\mathbf{A}}_G\right) = \frac{1}{K}\sum_{j=1}^{K}\cos\left(\frac{2ij\pi}{N}\right) \ \forall i = 0,1,\cdots,N-1.$$

We can then compute $|\cos(2\theta_w)|$ as $\max\{|\lambda(\hat{\mathbf{A}}_G)| < 1\}$.

2D PERIODIC GRID

The result for the 2D $m \times n$ periodic grid is equally straightforward to derive. The adjacency matrix of such a grid is equivalent to that of the Cartesian product of two ring graphs of length m and n respectively. From [110, Theorem 3] it follows that the eigenvalues of the adjacency matrix of this grid is given by

$$\lambda(\mathbf{A}_G) = \nu_{m,i} + \nu_{n,j} \ \forall i = 0,1,\cdots m-1, j = 0,1,\cdots,n-1,$$

where $\nu_{m,i}$ and $\nu_{n,j}$ are the ith and jth eigenvalues of the adjacency matrix of the length m and length n ring graph respectively. Given the d-regularity of the G, $\hat{\mathbf{A}}_G$ is simply a scaled version of this matrix. In this case, the scaling results in the final eigenvalues

$$\lambda\left(\hat{\mathbf{A}}_G\right) = \frac{1}{2}\left(\cos\left(\frac{2i\pi}{n}\right) + \cos\left(\frac{2j\pi}{m}\right)\right) \ \forall j = 0,1,\cdots m-1, k = 0,1,\cdots,n-1,$$

Overall, it follows that $|\cos(2\theta_w)| = \frac{1}{2}(\cos(\frac{2\pi}{n}) + \cos(\frac{2\pi}{m}))$.

5.6.3. FINITE TIME CONVERGENT PDMM

We now move to the final result of this paper, that of demonstrating an instance where PDMM converges in finite iterations. Specifically, from (5.23), for this to occur $|\delta| = |\cos(2\theta_w)| = 0$. The construction of such a finite convergence problem therefore requires both a certain network structure and a certain primal problem. In the case of the prior, from Table 5.1, the only networks considered therein which could achieve this point are the fully connected bipartite graphs as for all others $|\cos(2\theta_w)| \neq 0$. Furthermore, we know that any graph which is not d-regular will result in an increase of $|\delta|$ and thus cannot achieve finite convergence. Fortunately, by assuming N is even and setting $M = \frac{N}{2}$, it follows that such a network is both a fully connected bipartite graph and also d-regular.

The final piece of the puzzle is then to choose an appropriate primal problem. Notably, for $|\delta| = 0$, it follows from (5.32) that $\beta = \mu$. One such problem which satisfies this property is the average consensus problem

$$\min_{\mathbf{x}} \sum_{i\in V}\frac{1}{2}\|\mathbf{x}_i - \mathbf{a}_i\|^2 \quad \text{s.t.}\, \mathbf{x}_i = \mathbf{x}_j \quad \forall (i,j) \in E,$$

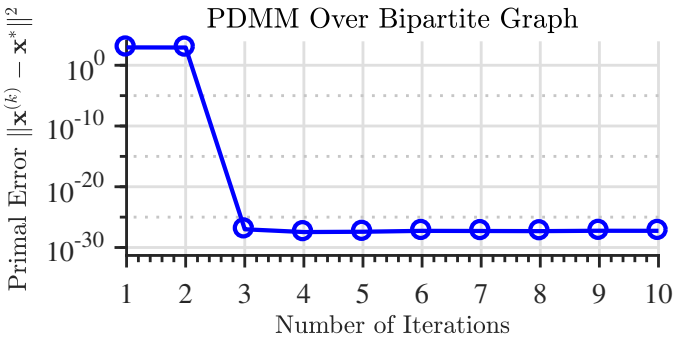


Figure 5.4: Average consensus over a complete bipartite graph with $M = \frac{N}{2}, \rho = \frac{2}{N}$ converges in exactly three iterations using PDMM regardless of the initialization used.

where $\beta = \mu = 1$ in this instance. By selecting $\rho = \frac{2}{N}$, as per (5.25), it follows from (5.32) that $|\delta| = 0$ as desired. Using the definition of $\mathbf{\Phi}$ in (5.19), we find that $\mathbf{\Phi}^2 = \mathbf{0}$ guaranteeing convergence of the iteration specific bound in only 2 steps.

The finite time convergence of PDMM is verified in Figure 5.4 for the case of a $N = 500$ network where a total of 1000 random starting vectors $\mathbf{z}^{(0)}$ were generated. The plot demonstrates the worst-case instance observed and shows that the primal variables of PDMM converge in exactly three iterations in this instance. While the auxiliary variables of PDMM converge in two iterations, a third primal update is required to compute the optimal $\mathbf{x}$ variables.

5.7. CONCLUSION

In this work we have explored the convergence rate of PDMM for strongly convex and smooth functions, and the role of network topology in this rate. The new convergence bound demonstrated herein is stronger than those currently in the literature and provides a tight bound on the limiting rate of PDMM for the considered functional class. For consensus problems, we have shown how this convergence is parameterized by the spectrum of the random walk matrix and used this insight to link the convergence of PDMM with random walks on graphs. Furthermore, for specific problem classes we have provided analytic expressions to understand how different commonly used topologies impact network convergence. Overall, this work provides insight into the effect of network topology in distributed optimization and highlights the importance of good network design for such applications.

APPENDICES

5.A. PROOF OF PROPOSITION 5.4.1

As ∇f is $\frac{1}{\beta}$-cocoercive, from Definition 5.4.5, $\forall \mathbf{x}_1, \mathbf{x}_2 \in \mathbf{X}$,

$$\left\langle \nabla f(\mathbf{x}_1) - \nabla f(\mathbf{x}_2), \mathbf{x}_1 - \mathbf{x}_2 \right\rangle \geq \frac{1}{\beta} \left\| \nabla f(\mathbf{x}_1) - \nabla f(\mathbf{x}_2) \right\|^2.$$

Combing this with (5.8) and (5.10), and defining $\mathbf{x}_a = \underset{\text{ran}(\mathbf{C}^T)}{\Pi} \mathbf{x}_1$ and $\mathbf{x}_b = \underset{\text{ran}(\mathbf{C}^T)}{\Pi} \mathbf{x}_2$, we find that, $\forall \mathbf{x}_a, \mathbf{x}_b \in \mathbf{X}_R$,

$$\left\langle \nabla g(\mathbf{x}_a) - \nabla g(\mathbf{x}_b), \mathbf{x}_a - \mathbf{x}_b \right\rangle \geq \frac{1}{\beta} \left\| \nabla g(\mathbf{x}_a) - \nabla g(\mathbf{x}_b) \right\|^2.$$

5.B. PROOF OF PROPOSITION 5.4.2

As ∇f is μ-strongly monotone, from Definition 5.4.3, $\forall \mathbf{x}_1, \mathbf{x}_2 \in \mathbf{X}$,

$$\left\langle \nabla f(\mathbf{x}_1) - \nabla f(\mathbf{x}_2), \mathbf{x}_1 - \mathbf{x}_2 \right\rangle \geq \mu \left\| \mathbf{x}_1 - \mathbf{x}_2 \right\|^2,$$

Combing this with (5.8) and (5.10), and defining $\mathbf{x}_a = \underset{\text{ran}(\mathbf{C}^T)}{\Pi} \mathbf{x}_1$ and $\mathbf{x}_b = \underset{\text{ran}(\mathbf{C}^T)}{\Pi} \mathbf{x}_2$, we find that, $\forall \mathbf{x}_a, \mathbf{x}_b \in \mathbf{X}_R$,

$$\left\langle \nabla g(\mathbf{x}_a) - \nabla g(\mathbf{x}_b), \mathbf{x}_a - \mathbf{x}_b \right\rangle \geq \mu \left\| \mathbf{x}_1 - \mathbf{x}_2 \right\|^2 \geq \mu \left\| \mathbf{x}_a - \mathbf{x}_b \right\|^2.$$

5.C. PROOF OF LEMMA 5.4.1

As g is μ-strong convex and β-smooth with respect to $\mathbf{X}_R$, $\mathbf{I} + \rho \frac{\mathbf{C}^{\dagger T}}{\rho} \nabla g \frac{\mathbf{C}^{\dagger}}{\rho}$ is $1 + \frac{\mu}{\rho \sigma^2_{\max}(C)}$-strongly monotone and $1 + \frac{\beta}{\rho \sigma^2_{\min \neq 0}(C)}$-Lipschitz continuous with respect to $\mathbf{X}_R$. Furthermore, $\mathbf{I} + \rho \frac{\mathbf{C}^{\dagger T}}{\rho} \nabla g \frac{\mathbf{C}^{\dagger}}{\rho} = \nabla \tilde{g}$ for a convex function $\tilde{g}$. It follows that the Fenchel conjugate function $\tilde{g}^*$ is $\hat{\beta}$-smooth and $\hat{\mu}$-strongly convex over ran $(\mathbf{C})$ where

$$\hat{\beta} = \frac{1}{1 + \frac{\mu}{\rho \sigma^2_{\max}(C)}}, \quad \hat{\mu} = \frac{1}{1 + \frac{\beta}{\rho \sigma^2_{\min \neq 0}(C)}}.$$

Combining Definition 5.4.2 for strong convexity and 5.4.4 for smoothness it follows that $\frac{(\hat{\beta} - \hat{\mu})}{2} \|\bullet\|^2 - \left(\tilde{g}^* - \frac{\hat{\mu}}{2} \|\bullet\|^2 \right)$ is convex such that $\tilde{g}^* - \frac{\hat{\mu}}{2} \|\bullet\|^2$ is $\hat{\beta} - \hat{\mu}$-smooth. As $\partial h^* = \partial h^{-1}$ for $h \in \Gamma_0$, it follows that $\nabla \tilde{g}^* - \hat{\mu} \mathbf{I} = \left(\mathbf{I} + \rho \frac{\mathbf{C}^{\dagger T}}{\rho} \nabla g \left(\frac{\mathbf{C}^{\dagger}}{\rho} \right) \right)^{-1} - \hat{\mu} \mathbf{I}$ is $\frac{1}{\hat{\beta} - \hat{\mu}}$ cocoercive. □

5.D. PROOF OF LEMMA 5.4.2

As the operator $\left(\mathbf{I}+\rho\frac{\mathbf{C}^{\dagger T}}{\rho}\nabla g\frac{\mathbf{C}^{\dagger}}{\rho}\right)^{-1}-\hat{\mu}\mathbf{I}$ is $\frac{1}{\hat{\beta}-\hat{\mu}}$ cocoercive it follows that the $\mathbf{x}$ iterates satisfy the inequality

$$\begin{aligned}&\left\langle \rho\mathbf{C}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^{*}\right)-\hat{\mu}\left(\mathbf{z}^{(k)}-\mathbf{z}^{*}\right),\underset{\text{ran}(\mathbf{C})}{\Pi}\left(\mathbf{z}^{(k)}-\mathbf{z}^{*}\right)\right\rangle\\&\geq\frac{1}{\hat{\beta}-\hat{\mu}}\left\|\rho\mathbf{C}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^{*}\right)-\hat{\mu}\underset{\text{ran}(\mathbf{C})}{\Pi}\left(\mathbf{z}^{(k)}-\mathbf{z}^{*}\right)\right\|^{2}.\end{aligned}$$

By then defining the vectors $\mathbf{v}_c=\frac{\hat{\beta}+\hat{\mu}}{2}\underset{\text{ran}(\mathbf{C})}{\Pi}\left(\mathbf{z}^{(k)}-\mathbf{z}^{*}\right)$, $\mathbf{v}_x=\rho\mathbf{C}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^{*}\right)$ and $\mathbf{v}_z=\underset{\text{ran}(\mathbf{C})}{\Pi}\left(\mathbf{z}^{(k)}-\mathbf{z}^{*}\right)$, it follows that we can rewrite this inequality as

$$\left\langle \mathbf{v}_x-\mathbf{v}_c+\frac{\hat{\beta}-\hat{\mu}}{2}\mathbf{v}_z,\mathbf{v}_z\right\rangle\geq\frac{1}{\hat{\beta}-\hat{\mu}}\left\|\mathbf{v}_x-\mathbf{v}_c+\frac{\hat{\beta}-\hat{\mu}}{2}\mathbf{v}_z\right\|^{2},$$

which is equivalent to the more simplistic form

$$\frac{1}{\hat{\beta}-\hat{\mu}}\left\langle \mathbf{v}_x-\mathbf{v}_c+\frac{\hat{\beta}-\hat{\mu}}{2}\mathbf{v}_z,-(\mathbf{v}_x-\mathbf{v}_c)+\frac{\hat{\beta}-\hat{\mu}}{2}\mathbf{v}_z\right\rangle\geq 0.$$

With some basic algebraic manipulation and by taking the square root of both sides of the inequality, we can finally recover the ellipsoidal constraint on the vector $\mathbf{v}_x$ given by

$$(\hat{\beta}-\hat{\mu})\left\|\mathbf{v}_z\right\|\geq\left\|\mathbf{v}_z-2\mathbf{v}_x-(\mathbf{v}_z-2\mathbf{v}_c)\right\|,$$

Using the definitions of $\mathbf{v}_c,\mathbf{v}_x$ and $\mathbf{v}_z$, it follows that

$$\begin{aligned}&(\hat{\beta}-\hat{\mu})\left\|\underset{\text{ran}(\mathbf{C})}{\Pi}\left(\mathbf{z}^{(k)}-\mathbf{z}^{*}\right)\right\|\geq\\&\left\|\underset{\text{ran}(\mathbf{C})}{\Pi}\left(\mathbf{y}^{(k+1)}-\mathbf{y}^{*}-(1-\hat{\beta}-\hat{\mu})\left(\mathbf{z}^{(k)}-\mathbf{z}^{*}\right)\right)\right\|\end{aligned}$$

completing the proof. □

III

DISTRIBUTED SOLVER DESIGN

6

A Distributed Algorithm for Separable Convex Optimization

Thomas Sherson, Richard Heusdens, and W. Bastiaan Kleijn

"Solving a problem for which you know there's an answer is like climbing a mountain with a guide, along a trail someone else has laid. In mathematics, the truth is somewhere out there in a place no one knows, beyond all the beaten paths. And it's not always at the top of the mountain. It might be in a crack on the smoothest cliff or somewhere deep in the valley."

Yoko Ogawa - *The Housekeeper and the Professor*

In this chapter we move to the task of broadening the class of problems which can be solved in a distributed manner. As our first contribution in this direction, we present a novel method for convex optimization in distributed networks called the distributed method of multipliers (DMM). The proposed method is based on a combination of a particular dual lifting and classic monotone operator splitting approaches to produce an algorithm with guaranteed asymptotic convergence in undirected networks. The proposed method allows any separable convex problem with linear constraints to be solved in undirected networks. In contrast to typical distributed approaches, the structure of the network does not restrict the types of problems which can be solved. Furthermore,

Parts of this chapter have been published in IEEE Transactions on Signal and Information Processing Over Networks, Accepted for Publication February 2019.

the solver can be applied to general separable problems, those with separable convex objectives and constraints, via the use of an additional primal lifting approach. Finally we demonstrate the use of DMM in solving a number of classic signal processing problems including beamforming, channel capacity maximization and portfolio optimization.

6.1. INTRODUCTION

Large scale optimization has become a significant topic of interest in the fields of computer science and electrical engineering. Driven by applications from the likes of the "Internet of Things" paradigm [5], cloud computing [111] and large scale machine learning [112], there is a growing need for efficient methods to solve large scale problems.

A family of approaches which has seen significant interest to address this need are those based on distributed computation. Designed for use in networked systems, distributed methods are often characterized by only requiring local computations at nodes within the network and short range communication. This removes the need for data aggregation and centralized computations which can quickly become infeasible to implement for large scale networks.

In recent years a wide range of techniques have been proposed to perform distributed computation including the likes of distributed consensus/gossip [10, 12, 72], belief propagation/message passing approaches [15, 14], graph signal processing over networks [73, 95, 17, 74] and more. In this work, we draw particular attention to the additional field of decentralized/distributed optimization as a method of achieving distributed computation. The motivation for this pursuit is the link between many signal processing applications and equivalent convex optimization problems [75]. By developing novel tools for distributed optimization we can facilitate a range of signal processing applications in a distributed manner.

While a number of distributed optimization algorithms exist within the literature, many were conceived for use in parallel computing rather than in-network applications [29]. The result is that while many algorithms allow for distribution over the structure of the target optimization problem they may require communication which does not respect the underlying network topology. As such, data aggregation approaches may be required which, as in the case of centralized computation, can be cumbersome or perhaps infeasible to implement. While the considered problem classes can be restricted to circumvent this point, this action in turn heavily limits their applicability to real world problems.

In this work we propose a novel method for distributed optimization which can be used to solve general *separable* convex optimization problems. The family of *separable* problems are characterized by having separable objectives functions and separable constraints. Unlike many existing approaches, for the proposed algorithm the underlying structure of the network need not affect the types of problems which can be addressed, allowing the solver to be readily applied to general undirected networked problems.

6.1.1. RELATED WORK

The field of parallel and distributed optimization has an extensive history upon which this paper builds. Key figures within the literature include Rockafellar, whose fundamen-

tal work on network optimization [76] and the relation between convex optimization and monotone operator theory [26, 27, 28] remains central to many results to this day. Notably, Rockafellar showed how linearly constrained convex programs with separable objectives could be solved in parallel via Lagrangian duality. This fundamental notion was further developed by the likes of Bertsekas, Tsitsiklis and Eckstein [29, 30, 31, 32, 86, 33] where again separability was used as a mechanism to design a range of distributed algorithms.

More recently, the demand for large scale data processing, has seen a return to form of these approaches within the literature [18, 37, 38, 79, 96]. This period has also seen the development of new methods of operator splitting [78, 77, 113] which in turn have motivated the development of further distributed optimization algorithms [19, 20, 97] again leveraging the fundamental links Rockafellar forged back in the 1970's. Unfortunately, a key limitations of many distributed algorithms is that they do not distribute in a way which respects the underlying network structure. This is highlighted in [53] where a distinction is drawn between the notions of distributability of an algorithm over the constraints and communication structure of a given networked optimization problem. The challenge is therefore to construct distributed algorithms that allow for simultaneous distribution over both the problem and network structure of a given application.

Within the literature, a number of approaches have been proposed to address the need for solving simultaneous distribution via varying means [52, 53, 54, 55, 56, 57, 58, 59]. In [55, 56], the alternating direction method of multipliers (ADMM) was used as a means of ensuring dual consensus, in turn guaranteeing primal optimality. Similarly, the methods in[52, 59] exploit dual decomposition based approaches, in combination with a consensus step and a proximal minimization step respectively to achieve the same feat. The work of [53] continues this trend by utilizing a combination of Lagrangian duality and an internal consensus algorithm (a gossip variant in this instance) to perform approximate dual updates. In contrast, the works of [57, 58, 54] utilize primal dual based techniques to tackle the presence of global constraints. Notably, the method in [54] aims to solve the more general problem of distributed saddle-point computation, a problem which includes distributed optimization as a special case. For distributed convex optimization, the proposed method reduces to a primal-dual sub-gradient algorithm incorporating a Laplacian averaging strategy.

6.1.2. MAIN CONTRIBUTIONS

The main contribution of this work is the proposal of a convex optimization solver termed the distributed method of multipliers (DMM) that is simultaneously distributable in both the network and problem structure. The proposed method is deployable in any undirected network topology so long as the network forms a single connected component. The result is an algorithm that respects the connectivity of the physical network whilst being applicable to a wide range of optimization problems.

The DMM algorithm is derived from the perspective of monotone operator theory and as such incorporates classic operator splitting approaches. This leads to a straightforward derivation closely related with other traditional algorithms from within the literature including the alternating direction method of multipliers (ADMM) [18], forward backward splitting (FB) [114] and more. The convergence guarantees of DMM follow

from its relation with Krasnosel'skiĭ-Mann iterations [115] and hold for all closed, convex and proper functions.

We demonstrate the use of the proposed method in practical signal processing problems including beamforming, channel capacity maximization and portfolio optimization. This numerically validates the performance claims of the algorithm while demonstrating how the approach can be deployed in practice.

6.1.3. ORGANIZATION OF PAPER

The remainder of this paper is organized as follows. In Section 6.2 we introduce basic nomenclature to support the remainder of the article. In Section 6.3 we derive our proposed distributed method for separable problems with affine constraints from a basic prototype optimization problem. Section 6.4 focuses on the computation of the iterates of our algorithm, demonstrating the efficiency and locality of the proposed method. Section 6.4.5 highlights a primal lifting stage, allowing the proposed method to also be used for general separable problems and provides a means for further reducing computational complexity. Section 6.5 demonstrates the use of the proposed method for a range of distributed signal processing tasks including beamforming, channel capacity maximization and portfolio optimization before making our concluding remarks in Section 6.6.

6.2. NOMENCLATURE

In this work we denote by $\mathbb{R}$ the set of real numbers, by $\mathbb{R}^N$ the set of real column vectors of length N and by $\mathbb{R}^{M\times N}$ the set of M by N real matrices. Let $\mathscr{X},\mathscr{Y}\subseteq\mathbb{R}^N$. A set valued operator $\mathbf{T}:\mathscr{X}\rightarrow\mathscr{Y}$ is defined by its graph, $\operatorname{gra}(\mathbf{T})=\{(\mathbf{x},\mathbf{y})\in\mathscr{X}\times\mathscr{Y}\mid\mathbf{y}\in\mathbf{T}(\mathbf{x})\}$. Similarly, the notion of an inverse of an operator $\mathbf{T}^{-1}$ is defined via its graph so that $\operatorname{gra}(\mathbf{T}^{-1})=\{(\mathbf{y},\mathbf{x})\in\mathscr{Y}\times\mathscr{X}\mid\mathbf{y}\in\mathbf{T}(\mathbf{x})\}$. $\mathbf{J}_{\mathbf{T},\rho}=(\mathbf{I}+\rho\mathbf{T})^{-1}$ denotes the resolvent of an operator while $\mathbf{R}_{\mathbf{T},\rho}=2\mathbf{J}_{\mathbf{T},\rho}-\mathbf{I}$ denotes the reflected resolvent (Cayley operator). The fixed-point set of $\mathbf{T}$ is denoted by $\operatorname{fix}(\mathbf{T})=\{\mathbf{x}\in\mathscr{X}\mid\mathbf{T}(\mathbf{x})=\mathbf{x}\}$.

6.3. DERIVING A DISTRIBUTED SOLVER FOR SEPARABLE CONVEX PROBLEMS WITH AFFINE CONSTRAINTS

In this section we introduce the derivation of the DMM algorithm for separable convex problems with affine constraints. In particular we demonstrate how DMM can be constructed via monotone operator splitting while respecting the underlying structure of a given physical network. This derivation also directly leads to convergence guarantees by linking the method with Krasnosel'skiĭ-Mann iterative schemes. This algorithm is extended to general separable problems in Section 6.4.5.

6.3.1. PROBLEM STATEMENT AND THE COMMUNICATION GRAPH

Consider an undirected network of compute nodes with which we want to perform distributed convex optimization. For now we make no assumptions on the structure of the network other than that it is simple, undirected and connected. In Section 6.3.5 we highlight the structural considerations of this network and how they can influence

the implementation of the algorithm. The communication structure of such a network can be represented by an equivalent *communication graph* which we denote by $G(V,E)$ where V is the set of nodes, $|V| = N$ is the number of nodes, $|\bullet|$ denotes the cardinality of a set and E denotes the set of undirected edges. These edges represent the physical communication channels between nodes.

An example of such a graph is included in Figure 6.1 for a simple eight node network. We denote by $\mathcal{N}(i) = \{j \in V \mid (i,j) \in E\}$ the neighborhood of node i, i.e. the set of nodes with which node i shares a physical connection. As an example, the neighborhood of node four in Figure 6.1 is given by the set $\mathcal{N}(4) = \{2,5,6\}$.

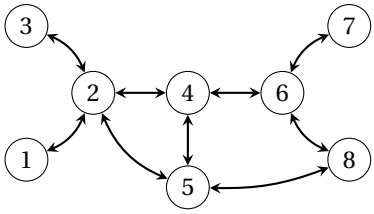


Figure 6.1: The communication graph G of a simple eight node network. Numbered circles denote nodes and their identifiers while the double ended arrows denote the undirected edges.

In this work, we are interested in using such a network to solve convex optimization problems in a distributed manner. For this section, we have restricted our attention to separable convex optimization problems with affine constraints. In these problems, each node is associated with a local objective function $f_i \in \Gamma_0\left(\mathbb{R}^{M_i}\right)$, $\forall i \in V$, parameterized by $\mathbf{x}_i \in \mathbb{R}^{M_i}$, where Γ_0 denotes the set of closed, proper and convex functions. We define the scalar $M_V = \sum_{i \in V} M_i$, which denotes the total number of variables in the network. More specifically, we consider problems of the form

$$\min_{\mathbf{x}_i} \sum_{i \in V} f_i(\mathbf{x}_i) \quad \text{s.t}\, \mathbf{A}_k \mathbf{x} - \mathbf{b}_k \underset{k}{\blacktriangleright} \mathbf{0} \quad \forall k \in \kappa \tag{6.1}$$

where for each k, $\underset{k}{\blacktriangleright}$ denotes either element-wise equality or inequality of the form $\geq$, $\kappa = \{1,2,\cdots,K\}$, K denotes the total number of constraints and $\mathbf{x} = \left[\mathbf{x}_1^T, \cdots, \mathbf{x}_N^T\right]^T$. The matrices $\mathbf{A}_k \in \mathbb{R}^{\mathscr{M}_k \times M_V}$ and vectors $\mathbf{b}_k \in \mathbb{R}^{\mathscr{M}_k}$ impose linear constraints between the variables at each node where $\mathscr{M}_k$ denotes the dimensionality of the kth constraint set. Importantly, we assume that (6.1) is strictly feasible such that strong duality holds.

Due to the separability of linear constraints, we can rewrite (6.1) by defining the sets V_k which denote those nodes i whose variables $\mathbf{x}_i$ play an active role in the kth constraint. More formally, for each $k \in \kappa$, this set is given by

$$V_k = \{i \in V \mid \mathbf{A}_{i,k} \neq \mathbf{0}\}.$$

Using this notation, (6.1) can be equivalently written as

$$\min_{\mathbf{x}_i} \sum_{i \in V} f_i(\mathbf{x}_i) \quad \text{s.t} \sum_{i \in V_k} \left(\mathbf{A}_{i,k}\mathbf{x}_i - \mathbf{b}_{i,k}\right) \underset{k}{\blacktriangleright} 0 \quad \forall k \in \kappa. \tag{6.2}$$

Here the matrices $\mathbf{A}_{i,k} \in \mathbb{R}^{\mathcal{M}_k \times M_i}$ are the ith set of columns of $\mathbf{A}_k$ such that $\mathbf{A}_k\mathbf{x} = \sum_{i \in V_k} \mathbf{A}_{i,k}\mathbf{x}_i$ while the vectors $\mathbf{b}_{i,k}$ are chosen such that $\sum_{i \in V_k} \mathbf{b}_{i,k} = \mathbf{b}_k$.

6.3.2. IMPLIED CONNECTIVITY OF THE CONSTRAINT GRAPH

The constraints in (6.2) imply a secondary set of relationships between the local variables at each node which can be modeled via a *constraint graph* denoted by $G_C(V, E_C)$. Here, the edge set E_C captures the interdependence of node variables in the constraint functions. In particular, if two nodes $i,\ j$ are active in the same constraint k, then $(i, j) \in E_C$. A fundamental challenge in distributed optimization follows from the differences between the edge sets of G_C and G as was highlighted in [53]. This challenge is best demonstrated with an example.

Consider again the network in Figure 6.1, equipped with the optimization problem

$$\begin{aligned} \min_{\mathbf{x}_i} \quad & \sum_{i=1}^{8} f_i(\mathbf{x}_i) \\ \text{s.t} \quad & \sum_{i=1,2,3} \left(\mathbf{A}_{i,1}^T \mathbf{x}_i - \mathbf{b}_{i,1}\right) = 0 \\ & \sum_{i=2,4,5,6} \left(\mathbf{A}_{i,2}^T \mathbf{x}_i - \mathbf{b}_{i,2}\right) = 0 \\ & \sum_{i=4,6,7,8} \left(\mathbf{A}_{i,3}^T \mathbf{x}_i - \mathbf{b}_{i,3}\right) = 0, \end{aligned} \tag{6.3}$$

where $\kappa = \{1,2,3\}$, $V_1 = \{1,2,3\}$, $V_2 = \{2,4,5,6\}$ and $V_3 = \{4,6,7,8\}$ in this instance. Note that for this example, we need not consider the dimensionality of the local variables, only the communication structure implied by the constraint set. Using the definition of E_C, we can form the constraint graph of (6.3) which is included in Figure 6.2.

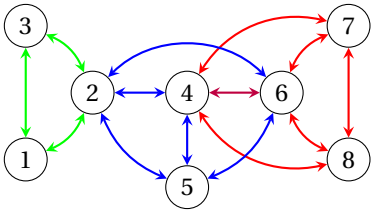


Figure 6.2: The constraint graph G_C for the same eight node network as in Figure 6.1. The green, blue and red colouration is used to denote the dependencies between nodes for first, second and third set of constraints. The *purple* edge between nodes four and six indicates the activity of both nodes in the red and blue constraint sets.

Comparing Figures 6.1 and 6.2, we can note that the connectivity of the physical network (G) and that imposed by the constraint functions (G_C) may differ depending on the optimization problem we are trying to solve. In particular, note the discrepancy between the edges of the E_C and E. The edge $(2,6)$, for instance, is contained with E_C but not in E. This poses a challenge for many existing algorithms which aim to distribute over the

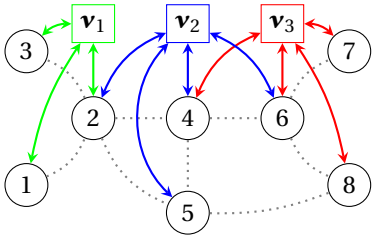


Figure 6.3: Graph of the dual problem for the same eight node network as in Figure 6.2. The green, blue and red colors denote the dependencies on the dual variables $\boldsymbol{\nu}_1$, $\boldsymbol{\nu}_2$ and $\boldsymbol{\nu}_3$.

constraint set [18, 96, 32] as while an algorithm may be distributable in G_C it may not be in G. Figure 6.2 for instance suggests the need for communication between nodes two and six which cannot be achieved in the physical network without relaying messages via node four. In the following we demonstrate a method to address this mismatch through a dual lifting approach.

6.3.3. Exploiting Separability Via Lagrange Duality

Inspired by classic results from Rockafellar [76], we can exploit the separability of (6.2) to overcome both the coupling of our primal variables through the constraint functions and thus the discrepancies between G_C and G. Specifically, we can use Lagrangian duality to rephrase (6.2) in an alternative form. For this purpose, at each node we define the set

$$\kappa_i = \{k \in \kappa | i \in V_k\},$$

to denote those k such that i is active in said constraints. Furthermore, the set of indices $k \in \kappa$ associated with inequality constraints in (6.2) is denoted by

$$\kappa_{\geq} = \left\{ k \in \kappa \mid \blacktriangleright_k \text{ is of the form} \geq \right\}.$$

The general form of the dual of (6.2) is therefore given by

$$\begin{aligned} \min_{\boldsymbol{\nu}_k} \quad & \sum_{i \in V} \left(f_i^* \left(\sum_{k \in \kappa_i} \mathbf{A}_{i,k}^T \boldsymbol{\nu}_k \right) - \sum_{k \in \kappa_i} \mathbf{b}_{i,k}^T \boldsymbol{\nu}_k \right) \\ \text{s.t.} \quad & \boldsymbol{\nu}_k \geq 0 \; \forall k \in \kappa_{\geq}, \end{aligned} \tag{6.4}$$

where $f_i^*(\mathbf{y}) = \sup_{\mathbf{x}} \left(\mathbf{y}^T \mathbf{x} - f(\mathbf{x}) \right)$ denotes the Fenchel conjugate of f_i and $\boldsymbol{\nu}_k \in \mathbb{R}^{\mathcal{M}_k}$ denotes the dual variable associated with the kth constraints. In the case of the graph considered in (6.2), a visualization of this point is provided in Figure 6.3. Here, the dual problem is parameterized by three dual variables ($\boldsymbol{\nu}_1$, $\boldsymbol{\nu}_2$ and $\boldsymbol{\nu}_3$), with each associated to a different constraint.

At this point many classic algorithms begin to directly solve (6.4) by distributing over the set of constraints. However, as each dual variable $\boldsymbol{\nu}$ can parameterize the local objective functions of multiple nodes, their updating can be challenging from a distributer perspective.

To demonstrate this challenge, consider a problem with a single constraint $k = 1$ in which every node plays an active note, i.e., $V_1 = V$. The resulting graphical model of the dual problem would exhibit a centralized topology in this instance. Such a topology implies the need for data aggregation to compute the dual variable in this instance which ultimately undermines the distributed intention of this work. For this reason, in the following we demonstrate a lifting approach which allows us to overcome this dual variable coupling while naturally respecting the underlying topology of the physical network.

6.3.4. A COMMUNICATION GRAPH PRESERVING DUAL LIFTING

Motivated by the dual lifting approach adopted in [99], we propose to address the coupling of the objectives functions by lifting the dimensionality of the dual problem. In particular, our objective is to rephrase (6.4) into a set of node and edge based terms. The proposed approach is referred to as the *extended dual* of Eq. (6.1) and is given by

$$\begin{aligned}
\min_{\boldsymbol{\lambda}_{i|j,k}} \quad & \sum_{i\in V}\left(f_i^*\left(\sum_{k\in\kappa_i}\sum_{j\in\mathcal{N}_k(i)}\frac{\mathbf{A}_{i,k}^T\boldsymbol{\lambda}_{i|j,k}}{|\mathcal{N}_k(i)|}\right)\right. \\
& \left.-\sum_{k\in\kappa_i}\sum_{j\in\mathcal{N}_k(i)}\frac{\mathbf{b}_{i,k}^T}{|\mathcal{N}_k(i)|}\boldsymbol{\lambda}_{i|j,k}\right) && \text{(6.5a)}\\
\text{s.t.} \quad & \boldsymbol{\lambda}_{i|j,k} = \boldsymbol{\lambda}_{j|i,k}\ \forall k\in\kappa, i\in V_k, j\in\mathcal{N}_k(i) && \text{(6.5b)}\\
& \boldsymbol{\lambda}_{i|j,k} = \boldsymbol{\lambda}_{i|l,k}\ \forall k\in\kappa, i\in V_k, j,l\in\mathcal{N}_k(i) && \text{(6.5c)}\\
& \boldsymbol{\lambda}_{i|j,k} \geq 0\ \forall k\in\kappa_{\geq}, i\in V_k, j\in\mathcal{N}_k(i). && \text{(6.5d)}
\end{aligned}$$

Here, $\mathcal{N}_k(i)$ denotes the *constrained neighborhood* of each node $i \in V_k$ where

$$\mathcal{N}_k(i) = \{j \in V_k \mid j \in \mathcal{N}(i)\},$$

i.e. the subset of $\mathcal{N}(i)$ active in the kth set of constraints.

To perform this dual lifting, $\forall k \in \kappa$ new copies of each dual variable $\boldsymbol{\nu}_k$ have been introduced for each directed edge $i|j \mid i, j \in V_k$ in the network. That is, $\forall k \in \kappa, i, j \in V_k$, the variables $\boldsymbol{\lambda}_{i|j,k}, \boldsymbol{\lambda}_{j|i,k} \in \mathbb{R}^{M_k}$ are introduced.

Equivalence with the original dual problem is insured via consensus constraints between dual variables corresponding to the same k. These can be divided into two types of constraints: edge based constraints of the form $\boldsymbol{\lambda}_{i|j,k} = \boldsymbol{\lambda}_{j|i,k}\ \forall k \in \kappa, (i,j) \in E$ and node based constraints $\boldsymbol{\lambda}_{i|j,k} = \boldsymbol{\lambda}_{i|l,k}\ \forall k \in \kappa, i \in V_k, j, l \in \mathcal{N}_k(i)$.

Performing the lifting in this way partitions the extended dual into four distinct sections: a fully node separable objective function (6.5a), a set of edge based consensus constraints (6.5b), an additional set of node based consensus constraints (6.5c) and finally a set of element-wise non-negativity constraints (6.5d). Such a problem structure is attractive in the context of alternating optimization methods as it partitions the problem into node and edge based terms.

For the example problem considered in (6.2), a visualization of the resulting lifted problem, indicating the relationship between the local copies of the dual variables, is included in Figure 6.4.

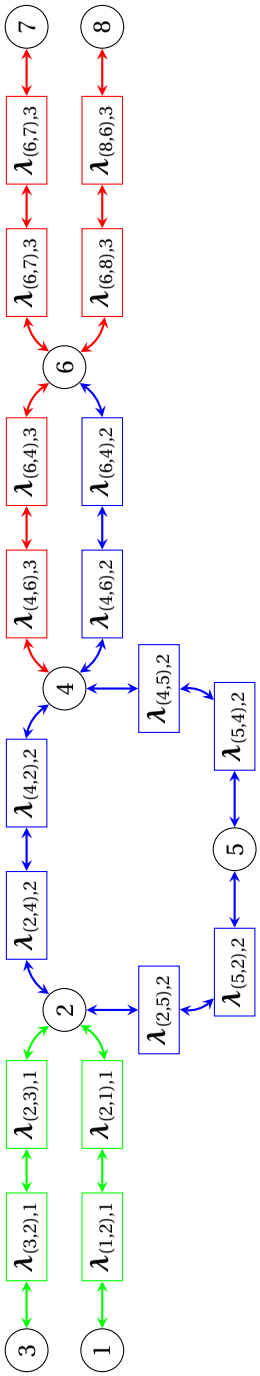


Figure 6.4: A graph modeling the connectivity of extended dual problem for the same eight node network as in Figure 6.2. The green, blue and red colors denote the dependencies on the three sets of constraints in (6.3).

Remark 2. *While the proposed lifting results in an increase in the dimensionality of the extended dual optimization problem, in Section 6.4, we demonstrate that this does not translate to an increase in the computational complexity of the local updates at each node. While not treated in this work, a more simplistic dual lifting such as introducing a single local version of $\boldsymbol{\nu}_k$ at each node $i \in V_k$, may not exhibit this favorable property, resulting in an increase in the computational complexity.*

6.3.5. NETWORK TOPOLOGY REQUIREMENTS

At this stage it is important to highlight the way in which the topology of G affects the feasibility of the lifting proposed in Eq. (6.5). In particular, the equivalence of Eq. (6.4) and (6.5) relies on the constraints enforcing consensus between dual variables $\forall k$. To this end, we demonstrate how this is guaranteed for a restricted set of network topologies which can then be generalized to the case of connected networks.

To begin, for the proposed lifting, a sufficient condition for the equivalence of Eq. (6.4) and (6.5) is given in Lemma 6.3.1.

Lemma 6.3.1. *If $\forall k \in \kappa$, the nodes $i \in V_k$ form a connected subgraph of G then (6.4) and (6.5) are equivalent problems.*

Proof. If $\forall k \in \kappa$, the set of nodes $i \in V_k$ form a connected subgraph of G then the constraints (6.5b) and (6.5c) ensure that $\exists\, \boldsymbol{\nu}_k$ such that at consensus, $\forall i \in V_k, j \in \mathcal{N}_k(i)$, $\boldsymbol{\lambda}_{i|j,k} = \boldsymbol{\nu}_k$. Hence the problems are equivalent. □

The sufficiency of this condition can be demonstrated via the example problem in Figures 6.1 and 6.2. Importantly, $\forall k \in \{1,2,3\}$ we can note that the active nodes form a connected subgraph of the underlying network G. This point is highlighted in Figure 6.5 where a distinction is drawn between the physical edges E (solid lines) and those required for a particular constraint set (dashed lines).

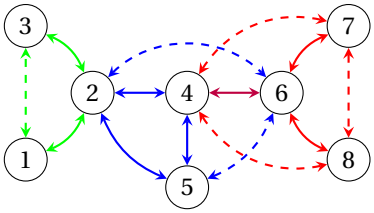
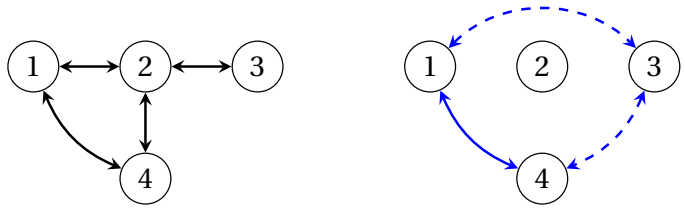


Figure 6.5: Constraint graph G_C for the eight node network as in Figure 6.2. Dashed line are used to identify those edges in E_C but not in E. $\forall k$, it can be noticed that the nodes $i \in V_k$ form a connected subgraph of G. As with Figure 6.2, the single purple edge between nodes four and six denotes the activity of both nodes in the red and blue constraint sets.

While sufficient to guarantee equivalence, Lemma 6.3.1 seems restrictive. For instance, if a network forms a single connected component, data aggregation could be used to enforce dual consensus without satisfying this condition. As a demonstration, consider the communication graph and constraint graph given in Figure 6.6a and 6.6b



(a) Communication Graph (b) Constraint Graph

Figure 6.6: An example of a simple four node network. The target problem of interest imposes no constraints on the primal variables at node 2 but does affect the other three nodes. In contrast to Figure 6.5, the set of active nodes do not form a connected subgraph of G.

respectively. Clearly, the network has the physical connectivity to enforce any set of constraints between nodes but, due to the lack of activity of node two in the constraints, the set of active nodes forms a physically disjoint subgraph.

To generalize the class of applicable networks we can introduce a modification to the dual lifting in (6.4) and (6.5) to ensure its equivalence to (6.6). The basic notion is that $\forall k \in \kappa$ we can introduce additional nodes to the set V_k such that the resulting subnetworks form connected subgraphs. This action can always be performed due to our initial assumption that G forms a single connected component.

In the case of the networked problem in Figure 6.6, the initial constraint set $V_k = \{1,3,4\}$ can be augmented to include node two such that $V_k = \{1,2,3,4\}$ which in turn ensures that the constraint subgraph forms a single connected component. This introduces local copies of $\boldsymbol{\nu}_k$ at node two denoted by $\boldsymbol{\lambda}_{(2,j),k}$ $\forall j \in \mathcal{N}_k(2)$. These additional variables in no way influence the objective cost of node two and exist only to enforce consensus between the lifted dual variables $\forall i \in V_k$. The additional matrix $\mathbf{A}_{2,k} = \mathbf{0}$ and vector $\mathbf{b}_{2,k} = \mathbf{0}$ are also introduced to complete the modification. For the remainder of the document, should a network require this modification to solve a particular problem we assume that this is performed.

6.3.6. SIMPLIFYING THE PROBLEM NOTATION

To assist with the remainder of this derivation, we introduce a compact notation to allow us to simplify Eq. (6.5). In particular, we show that (6.5) can be rewritten as

$$\begin{aligned} \min_{\boldsymbol{\lambda}} \quad & f^*(\mathbf{C}^T\boldsymbol{\lambda}) - \mathbf{d}^T\boldsymbol{\lambda} \\ \text{s.t.} \quad & (\mathbf{I}-\mathbf{P})\,\boldsymbol{\lambda} = \mathbf{0},\ \mathscr{L}\boldsymbol{\lambda} = \mathbf{0},\ \mathbf{S}\boldsymbol{\lambda} \geq \mathbf{0}, \end{aligned} \tag{6.6}$$

where the three constraints correspond to (6.5b), (6.5c) and (6.5d) respectively. The additional matrices associated with this equivalent representation are defined below.

FORMING A SINGLE DUAL VECTOR

We firstly define a vector notation for the extended dual variables. For each $k \in \kappa$, denote by $\boldsymbol{\lambda}_k$ the stacked vector of all $\boldsymbol{\lambda}_{i|j,k}$. The ordering of this stacking is based on the directed edge index and is given by $(1,2) < (1,3) < \cdots < (1,N) < (2,1) < (2,3) < \cdots <$

$(N, N-1)$. In this way, $\boldsymbol{\lambda}_k$ is given by

$$\boldsymbol{\lambda}_k = \left[\boldsymbol{\lambda}_{(1,2),k}, \cdots, \boldsymbol{\lambda}_{(1,N),k}, \boldsymbol{\lambda}_{(2,1),k}, \cdots, \boldsymbol{\lambda}_{(N,N-1),k}\right]^T.$$

By stacking the set of all $\boldsymbol{\lambda}_k$, we can then form a single dual vector $\boldsymbol{\lambda}$ as used in (6.6) such that

$$\boldsymbol{\lambda} = \left[\boldsymbol{\lambda}_1^T, \cdots, \boldsymbol{\lambda}_K^T\right]^T.$$

COMPACT OBJECTIVE NOTATION

Given the compact notation of the dual variables, we now move to simplifying the objective function. Firstly, we define the global function

$$\begin{aligned} f : \mathbb{R}^{M_V} &\mapsto \mathbb{R} \\ \mathbf{x} &\mapsto \sum_{i \in V} f_i(\mathbf{x}_i) \end{aligned}$$

where $\mathbb{R}^{M_V} = \mathbb{R}^{M_1} \times \ldots \times \mathbb{R}^{M_N}$. Similarly, the Fenchel conjugate of this function is denoted by f^*.

The next step is to define a matrix and vector to rewrite our objective using $\boldsymbol{\lambda}$ and f^*. This stage is broken into multiple steps. Firstly, $\forall k \in \kappa, i \in V_k$ we define $\mathbf{C}_{i,k}$ and $\mathbf{d}_{i,k}$ as

$$\begin{aligned} \mathbf{C}_{i,k} &= \mathbf{1}_{|\mathcal{N}_k(i)|} \otimes \frac{\mathbf{A}_{i,k}^T}{|\mathcal{N}_k(i)|} \; \forall i \in V_k, \\ \mathbf{d}_{i,k} &= \mathbf{1}_{|\mathcal{N}_k(i)|} \otimes \frac{\mathbf{b}_{i,k}}{|\mathcal{N}_k(i)|} \; \forall i \in V_k, \end{aligned}$$

where $\otimes$ denotes the Kronecker product of two matrices and the notation $\mathbf{1}_{|\mathcal{N}_k(i)|}$ is used to indicate a $|\mathcal{N}_k(i)|$-length column vector of ones. For each $k \in \kappa$ we therefore define the matrices $\mathbf{C}_k$ and $\mathbf{d}_k$ as

$$\mathbf{C}_k = \begin{bmatrix} \mathbf{C}_{1,k} & \ldots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \ldots & \mathbf{C}_{N,k} \end{bmatrix}, \; \mathbf{d}_k = \left[\mathbf{d}_{1,k}^T, \cdots, \mathbf{d}_{N,k}^T\right]^T. \tag{6.7}$$

We can then form the final matrix $\mathbf{C}$ and vector $\mathbf{d}$ by stacking over the set of constraints so that

$$\begin{aligned} \mathbf{C} &= \left[\mathbf{C}_1^T, \cdots, \mathbf{C}_K^T\right]^T, \\ \mathbf{d} &= \left[\mathbf{d}_1^T, \cdots, \mathbf{d}_K^T\right]^T. \end{aligned}$$

Combining these two definitions the objective function of (6.5) can be compactly written as

$$f^*(\mathbf{C}^T\boldsymbol{\lambda}) - \mathbf{d}^T\boldsymbol{\lambda}.$$

Compact Constraint Notation

As with the objective, we can define a set of additional matrices to rewrite the constraints using our dual vector notation. To capture the edge based constraints (6.5b), we define for each $k \in \kappa$ the symmetric permutation matrix $\mathbf{P}_k$ which interchanges the edge variables $\boldsymbol{\lambda}_{i|j,k}, \boldsymbol{\lambda}_{j|i,k}\ \forall i, j \in V_k$. By concatenating over all $k \in \kappa$, we can define the permutation matrix

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_1 & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \mathbf{P}_K \end{bmatrix}.$$

Combining with the definition of $\boldsymbol{\lambda}$ allows us to rewrite (6.5b) as $(\mathbf{I} - \mathbf{P})\,\boldsymbol{\lambda} = \mathbf{0}$ to enforce edge based consensus.

For the second set of constraints (6.5c), we define the matrices

$$\mathscr{L}_k = \begin{bmatrix} \mathscr{L}_{1,k} & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \mathscr{L}_{N,k} \end{bmatrix} \otimes \mathbf{I}_{\mathscr{M}_k} \quad \forall k \in \kappa,$$

where

$$\begin{aligned} \mathscr{L}_{i,k} &= \mathbf{D}_{i,k} - \mathbf{E}_{i,k}\ \forall i \in V_k \\ \mathbf{D}_{i,k} &= (|\mathscr{N}_k(i)| - 1)\,\mathbf{I}_{|\mathscr{N}_k(i)|}\ \forall i \in V_k \\ \mathbf{E}_{i,k} &= \mathbf{1}_{|\mathscr{N}_k(i)|}\mathbf{1}^T_{|\mathscr{N}_k(i)|} - \mathbf{I}_{|\mathscr{N}(i)|}\ \forall i \in V_k. \end{aligned}$$

Similar to $\mathbf{1}_M$, here the matrix $\mathbf{I}_M$ is used to denote an $M \times M$ identity matrix. The matrix $\mathscr{L}_k$ can be thought of as a block diagonal matrix of graph Laplacians. Importantly, it can be shown that $\forall k \in \kappa, i \in V_k$

$$\begin{aligned} \mathscr{L}_{i,k}\mathbf{1}_{|\mathscr{N}_k(i)|} &= \left(\mathbf{D}_{i,k} - \mathbf{E}_{i,k}\right)\mathbf{1}_{|\mathscr{N}_k(i)|} \\ &= |\mathscr{N}_k(i)|\mathbf{1}_{|\mathscr{N}_k(i)|} - |\mathscr{N}_k(i)|\mathbf{1}_{|\mathscr{N}_k(i)|} \\ &= \mathbf{0}_{|\mathscr{N}_k(i)|}, \end{aligned}$$

where in the third line we have used the mixed-product property of Kronecker products. In this way, the kernel space of $\mathscr{L}_{i,k}$ corresponds to the consensus vector and can therefore be used to impose the consensus constraints in (6.5c). Concatenating over the constraints, we can form the matrix

$$\mathscr{L} = \begin{bmatrix} \mathscr{L}_1 & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \mathscr{L}_K \end{bmatrix},$$

such that (6.5c) can be rewritten as $\mathscr{L}\boldsymbol{\lambda} = \mathbf{0}$. Furthermore, from the structure in (6.7), $\mathscr{L}_{i,k}\mathbf{C}_{i,k} = \mathbf{0}$, $\mathscr{L}_{i,k}\mathbf{d}_{i,k} = \mathbf{0}$, $\forall k \in \kappa, i \in V_k$ such that $\mathscr{L}\mathbf{C} = \mathbf{0}$, $\mathscr{L}\mathbf{d} = \mathbf{0}$.

For the final set of constraints (6.5d), $\forall k \in \kappa$ we define the selection matrices $\mathbf{S}_k \in \mathbb{R}^{\mathscr{M}_k \times \mathscr{M}_k}$ given by

$$\mathbf{S}_k = s_k \mathbf{I}_{\mathscr{M}_k \sum_{i \in V_k} |\mathscr{N}_k(i)|},\ s_k = \begin{cases} 1 \text{ if } k \in \kappa_{\geq} \\ 0 \text{ otherwise,} \end{cases}$$

which preserves those dual variables associated with the inequality constraints. Concatenating over the constraints, we can form the final selection matrix given by

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}_1 & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \mathbf{S}_K \end{bmatrix},$$

such that (6.5d) can be rewritten as $\mathbf{S}\boldsymbol{\lambda} \geq \mathbf{0}$.

Using this compact notation, we are now ready to define our proposed distributed optimization algorithm.

6.3.7. FROM THE EXTENDED DUAL PROBLEM TO A MONOTONIC INCLUSION

Given the lifted dual problem (6.6), we now move to defining a distributed algorithm to compute an optimizer of (6.1). In particular, we want to construct an iterative algorithm

$$\mathbf{y}^{(t+1)} = \mathbf{U}_E \circ \mathbf{U}_V \left(\mathbf{y}^{(t)}\right), \tag{6.8}$$

which converges to a minimizer of (6.1) where t indicates the iteration number, $\mathbf{y} \in \mathbb{R}^p$ are the variables of interest and the operators $\mathbf{U}_E : \mathbb{R}^p \mapsto \mathbb{R}^p$ are $\mathbf{U}_V : \mathbb{R}^p \mapsto \mathbb{R}^p$ are parallelizable over the nodes and edges respectively. The additional notation $\circ$ is used to denote operator composition so that $\forall\, (\mathbf{x},\mathbf{z}) \in \text{gra}\,(\mathbf{S}_1 \circ \mathbf{S}_2)$, $\exists \mathbf{y} \mid \left(\mathbf{x},\mathbf{y}\right) \in \text{gra}\,(\mathbf{S}_1), \left(\mathbf{y},\mathbf{z}\right) \in \text{gra}\,(\mathbf{S}_2)$. We would like such operators to be at least nonexpansive so that classic iterative solvers can be employed. The nonexpansiveness of an operator is defined as follows.

Definition 6.3.1. *Nonexpansive Operators: An operator* $\mathbf{T} : \mathscr{X} \to \mathscr{Y}$ *is nonexpansive if*

$$\|\mathbf{u}-\mathbf{v}\| \leq \|\mathbf{x}-\mathbf{y}\| \quad (\mathbf{x},\mathbf{u}), \left(\mathbf{y},\mathbf{v}\right) \in \text{gra}\,(\mathbf{T}),$$

where $\langle \mathbf{x},\mathbf{y} \rangle$ *denotes the inner product between* $\mathbf{x} \in \mathscr{X}$ *and* $\mathbf{y} \in \mathscr{Y}$ *and* $\|\mathbf{x}\|$ *denotes the associated induced norm.*

We can construct an iterative solver for (6.1) via classic operator splitting approaches. In this case, we make use of the relationship between monotone operators and the subdifferentials of convex functions. In particular, an operator is monotone if it satisfies the following definition:

Definition 6.3.2. *Monotone Operators: An operator* $\mathbf{T} : \mathscr{X} \to \mathscr{Y}$ *is monotone iff*

$$\langle \mathbf{u}-\mathbf{v}, \mathbf{x}-\mathbf{y} \rangle \geq 0 \quad \forall\, (\mathbf{x},\mathbf{u}), \left(\mathbf{y},\mathbf{v}\right) \in \text{gra}\,(\mathbf{T}),$$

Furthermore, $\mathbf{T}$ *is maximal monotone iff*

$$\nexists \text{ a monotone } \tilde{\mathbf{T}} : \mathscr{X} \to \mathscr{Y} \mid \text{gra}(\mathbf{T}) \subset \text{gra}(\tilde{\mathbf{T}}).$$

To form our iterative approach, consider the equivalent unconstrained form of (6.6),

$$\min_{\boldsymbol{\lambda}} \quad f^*(\mathbf{C}^T\boldsymbol{\lambda}) - \mathbf{d}^T\boldsymbol{\lambda} + \iota_{\mathscr{C}_1}(\boldsymbol{\lambda}) + \iota_{\mathscr{C}_2}(\boldsymbol{\lambda}) + \iota_{\mathscr{C}_3}(\boldsymbol{\lambda}), \tag{6.9}$$

where $\iota_{\mathscr{C}}$ denotes an indicator function of the set $\mathscr{C}$ such that

$$\iota_{\mathscr{C}}(\boldsymbol{\lambda}) = \begin{cases} 0 & \text{if } \boldsymbol{\lambda} \in \mathscr{C} \\ +\infty & \text{otherwise} \end{cases}.$$

Here the convex sets $\mathscr{C}_1$, $\mathscr{C}_2$ and $\mathscr{C}_3$ are given by

$$\mathscr{C}_1 = \ker(\mathbf{I} - \mathbf{P}), \ \mathscr{C}_2 = \ker(\mathscr{L}), \ \mathscr{C}_3 = \{\boldsymbol{\lambda} \mid \mathbf{S}\boldsymbol{\lambda} \geq \mathbf{0}\},$$

where $\ker(\mathscr{C})$ denotes the kernel of $\mathscr{C}$.

As $f \in \Gamma_0$, as the sets $\mathscr{C}_1$, $\mathscr{C}_2$ and $\mathscr{C}_3$ are closed subspaces such that $\iota_{\mathscr{C}_1}(\boldsymbol{\lambda})$, $\iota_{\mathscr{C}_2}(\boldsymbol{\lambda})$, $\iota_{\mathscr{C}_3}(\boldsymbol{\lambda}) \in \Gamma_0$, and as all functions contain a common feasible point, a minimizer of (6.9) can be found by finding a zero-point of its subdifferential.

$$\begin{aligned} \mathbf{0} \in \mathbf{C}\partial f^*\left(\mathbf{C}^T\boldsymbol{\lambda}\right) - \mathbf{d} + \partial\iota_{\mathscr{C}_1}(\boldsymbol{\lambda}) \\ + \partial\iota_{\mathscr{C}_2}(\boldsymbol{\lambda}) + \partial\iota_{\mathscr{C}_3}(\boldsymbol{\lambda}). \end{aligned} \tag{6.10}$$

Here ∂f is used to denotes the subdifferential of f.

A zero-point of (6.10) can be found via a range of approaches including Forward Backward (FB) splitting, Douglas-Rachford (DR) Splitting, Primal-Dual Hybrid Gradient (PDHG) (Chambolle Pock) and more (see [47] for an overview of such splitting methods). In the proposed distributed context however, the choice of a such splitting method must be made to take advantage of the node and edge based structure we have introduced into the subdifferentials of (6.10).

In this work, we adopt a classic two operator splitting scheme to rephrase (6.10) as a more familiar fixed point inclusion. To do so, we define the two operators

$$\mathbf{T}_1(\boldsymbol{\lambda}) = \mathbf{C}\partial f^*\left(\mathbf{C}^T\boldsymbol{\lambda}\right) - \mathbf{d} + \partial\iota_{\mathscr{C}_2}(\boldsymbol{\lambda}), \tag{6.11a}$$

$$\mathbf{T}_2(\boldsymbol{\lambda}) = \partial\iota_{\mathscr{C}_1}(\boldsymbol{\lambda}) + \partial\iota_{\mathscr{C}_3}(\boldsymbol{\lambda}), \tag{6.11b}$$

where, by design, $\mathbf{T}_1$ is node separable and $\mathbf{T}_2$ is edge separable. These operators are both maximal monotone by combining the results of [28] and [116].

6.3.8. Operator Splitting Via Peaceman-Rachford Splitting

To find a minimizer of the extended dual problem, we can use PR splitting to recast (6.10) as a fixed point problem of a nonexpansive operator. Our motivation for choosing this approach is that, unlike methods such as FB splitting, we need not impose additional functional restrictions beyond that $f \in \Gamma_0$. Additionally, PR splitting allows us to take advantage of the node and edge separability of $\mathbf{T}_1$ and $\mathbf{T}_2$ respectively. In contrast, other methods such as PDHG cannot take advantage of this point. In particular, PDHG would require that either $\mathbf{T}_1$ or $\mathbf{T}_2$ could be expressed as a composition of a subdifferential and a linear operator. While we could define an alternative operator $\hat{\mathbf{T}}_1 = \mathbf{C}\partial f^*\left(\mathbf{C}^T\boldsymbol{\lambda}\right) - \mathbf{d}$ such that this method could be used, the second operator would be given by $\hat{\mathbf{T}}_2(\boldsymbol{\lambda}) = \partial\iota_{\mathscr{C}_1}(\boldsymbol{\lambda}) + \partial\iota_{\mathscr{C}_3}(\boldsymbol{\lambda}) + \partial\iota_{\mathscr{C}_2}(\boldsymbol{\lambda})$ which is neither edge or node separable, eliminating our ability to form a distributed solver. In this way, PR splitting was a natural choice.

6.3.9. FORMING THE DISTRIBUTED METHOD OF MULTIPLIERS

Given two maximal monotone operators $\mathscr{A}$ and $\mathscr{B}$ and a positive scalar $\rho > 0$, PR splitting can be used to find a zero of $\mathscr{A} + \mathscr{B}$ by rephrasing it as a more familiar fixed point condition [47, Sec. 7.3] of the form

$$\mathbf{R}_{\mathscr{B},\rho} \circ \mathbf{R}_{\mathscr{A},\rho}(\mathbf{z}) \in \mathbf{z},\ \boldsymbol{\lambda} = \mathbf{J}_{\mathscr{A},\rho}(\mathbf{z}), \tag{6.12}$$

where $\mathbf{J}_{\mathscr{A},\rho}$ and $\mathbf{R}_{\mathscr{A},\rho}$ denote the resolvent and reflected resolvent of $\mathscr{A}$ respectively. The newly introduced $\mathbf{z}$ variables will be referred to as *auxiliary* variables from here on out. In the particular case of (6.10), we can therefore form the fixed point condition

$$\mathbf{R}_{\mathbf{T}_2,\rho} \circ \mathbf{R}_{\mathbf{T}_1,\rho}(\mathbf{z}) \in \mathbf{z},\ \boldsymbol{\lambda} = \mathbf{J}_{\mathbf{T}_1,\rho}(\mathbf{z}). \tag{6.13}$$

As we will show in the coming section, the node and edge separable structure of $\mathbf{T}_1$ and $\mathbf{T}_2$, is inherited by the operators $\mathbf{J}_{\mathbf{T}_1,\rho}$ and $\mathbf{J}_{\mathbf{T}_2,\rho}$ respectively so that $\mathbf{R}_{\mathbf{T}_1,\rho}$ and $\mathbf{R}_{\mathbf{T}_2,\rho}$ form the operators $\mathbf{U}_V$ and $\mathbf{U}_E$ outlined in Eq. (6.8).

To form our distributed algorithm, we define the nonexpansive distributed method of multipliers (DMM) operator as $\mathbf{T}_{D,\rho} = \mathbf{R}_{\mathbf{T}_2,\rho} \circ \mathbf{R}_{\mathbf{T}_1,\rho}$. The nonexpansiveness here stems from the maximal monotonicity of $\mathbf{T}_1$ and $\mathbf{T}_2$ and thus the nonexpansiveness of $\mathbf{R}_{\mathbf{T}_1,\rho}$ and $\mathbf{R}_{\mathbf{T}_2,\rho}$. As the composition of nonexpansive operators, it follows that $\mathbf{T}_{D,\rho}$ is also nonexpansive.

Remark 3. *In the specific case that all the constraints are edge based (only two nodes are active in each constraint and they correspond to a physical edge of G) the DMM operator corresponds to the PDMM operator given in [99].*

Given the nonexpansiveness of $\mathbf{T}_{D,\rho}$, we can employ a Krasnosel'skiĭ-Mann type iterative scheme to find a fixed point of the operator [115]. Such a scheme is given by

$$\mathbf{z}^{(t+1)} = (1-\alpha^{(t)})\mathbf{z}^{(t)} + \alpha^{(t)}\mathbf{T}_{D,\rho}\left(\mathbf{z}^{(t)}\right). \tag{6.14}$$

where $\forall\, t \in \mathbb{N}, \alpha^{(t)} \in (0,1)$ and the sequence of $\alpha^{(t)}$ is non-convergent i.e. $\sum_{t=0}^{+\infty} \alpha^{(t)} = +\infty$. As a special case of the general Krasnosel'skiĭ-Mann scheme, when $\alpha^{(t)} = \frac{1}{2}\ \forall\, t \in \mathbb{N}$, we recover the Douglas-Rachford splitting variant of the DMM algorithm. Such an approach is closely related to the well known alternating direction method of multipliers (ADMM).

6.4. COMPUTATION OF THE DMM UPDATE EQUATIONS

Given the basic iterative scheme for DMM, presented in (6.14), in this section we demonstrate how the structure of (6.2) can be used to simplify the computation of the iterates. This is comprised of two simplifications, one for each of the reflected resolvents, and is summarized in the following two Lemmas.

6.4.1. COMPUTING THE REFLECTED RESOLVENT $\mathbf{R}_{\mathbf{T}_1,\rho}$

We begin with the first reflected resolvent operator $\mathbf{R}_{\mathbf{T}_1,\rho}$ and its method of computation.

Lemma 6.4.1.

$$\begin{aligned}\mathbf{R}_{\mathbf{T}_1,\rho}\left(\mathbf{z}^{(t)}\right) &= \mathbf{w}^{(t+1)} = 2\boldsymbol{\lambda}^{(t+1)} - \mathbf{z}^{(t)} \\ &= 2\boldsymbol{\gamma}^{(t+1)} - 2\rho\left(\mathbf{C}\mathbf{x}^{(t+1)} - \mathbf{d}\right) - \mathbf{z}^{(t)},\end{aligned}$$

where,

$$\begin{aligned}\boldsymbol{\lambda}^{(t+1)} &= \mathbf{J}_{\mathbf{T}_1,\rho}\left(\mathbf{z}^{(t)}\right) \text{ by (6.13)}, \boldsymbol{\gamma}^{(t+1)} = \prod_{\ker(\mathscr{L})}\left(\mathbf{z}^{(t)}\right), \\ \mathbf{x}^{(t+1)} &= \arg\min_{\mathbf{x}}\left(f(\mathbf{x}) + \frac{\rho}{2}\|\mathbf{C}\mathbf{x} - \mathbf{d}\|^2 - \langle \mathbf{C}^T\boldsymbol{\gamma}^{(t+1)}, \mathbf{x}\rangle\right).\end{aligned}$$

The introduced $\mathbf{x}^{(t+1)}$ *iterates are estimates of the primal minimizers of* (6.1).

The proof for this Lemma can be found in Appendix 6.A.

Note that the primal update, which involves a convex optimization problem, has local variables of the same dimensionality as the original problem in (6.1). If an alternative lifting were utilized, the increase in the number of local variables at each node may unnecessarily result in a more complex local optimization problem per iteration.

6.4.2. COMPUTING THE REFLECTED RESOLVENT $\mathbf{R}_{\mathbf{T}_2,\rho}$

In the case of $\mathbf{T}_2$, it can be shown that the reflected resolvent $\mathbf{R}_{\mathbf{T}_2,\rho}$ also exhibits a naturally distributable solution.

Lemma 6.4.2.

$$\begin{aligned}\mathbf{R}_{\rho,\mathbf{T}_2}\left(\mathbf{w}^{(t+1)}\right) &= \mathbf{v}^{(t+1)} = 2\mathbf{y}^{(t+1)} - \mathbf{w}^{(t+1)} \\ &= \mathbf{P}\mathbf{w}^{(t+1)} - \min\{\mathbf{S}(\mathbf{I}+\mathbf{P})\mathbf{w}^{(t+1)}, \mathbf{0}\},\end{aligned}$$

where

$$\begin{aligned}\mathbf{y}^{(t+\frac{1}{2})} &= \frac{1}{2}(\mathbf{I}+\mathbf{P})\mathbf{w}^{(t+1)}, \\ \mathbf{y}^{(t+1)} &= \mathbf{y}^{(t+\frac{1}{2})} - \min\{\mathbf{S}\mathbf{y}^{(t+\frac{1}{2})}, \mathbf{0}\}.\end{aligned}$$

and $\min\{\bullet,\bullet\}$ *is used to denote elementwise minimization.*

The proof for this Lemma can be found in Appendix 6.B. $\mathbf{R}_{\rho,\mathbf{T}_2}$ reduces to a local exchanging of information between neighboring nodes (indicated by the use of the permutation operation) followed by a localized post processing at each node comprised of linear operations and element-wise comparisons.

6.4.3. IMPLEMENTATION IN A DISTRIBUTED NETWORK

By combining Lemmas 6.4.1 and 6.4.2, the DMM algorithm can be expressed as

$$\boldsymbol{\gamma}^{(t+1)} = \prod_{\ker(\mathscr{L})} \mathbf{z}^{(t)} \tag{6.15a}$$

$$\mathbf{x}^{(t+1)} = \arg\min_{\mathbf{x}} \Big(f(\mathbf{x}) - \langle \mathbf{C}^T \boldsymbol{\gamma}^{(t+1)}, \mathbf{x} \rangle + \frac{\rho}{2} \|\mathbf{C}\mathbf{x} - \mathbf{d}\|^2 \Big) \tag{6.15b}$$

$$\mathbf{w}^{(t+1)} = 2\boldsymbol{\gamma}^{(t+1)} - 2\rho\left(\mathbf{C}\mathbf{x}^{(t+1)} - \mathbf{d}\right) - \mathbf{z}^{(t)} \tag{6.15c}$$

$$\mathbf{v}^{(t+1)} = \mathbf{P}\mathbf{w}^{(t+1)} - \min\{\mathbf{S}(\mathbf{I}+\mathbf{P})\mathbf{w}^{(t+1)}, \mathbf{0}\} \tag{6.15d}$$

$$\mathbf{z}^{(t+1)} = \left(1-\alpha^{(t)}\right)\mathbf{z}^{(t)} + \alpha^{(t)}\mathbf{v}^{(t+1)} \tag{6.15e}$$

The computation of each iteration reduces to a local averaging step at each node (6.15a), a single optimization over the primal variables (6.15b), the sharing of data between neighboring nodes (6.15d) and a set of additional matrix vector multiplications and element-wise comparisons. Furthermore, all of these operations are inherently distributable within the original network with (6.15a), (6.15b) and (6.15c) corresponding to $\mathbf{R}_{\mathbf{T}_1,\rho}$ and (6.15d) to $\mathbf{R}_{\mathbf{T}_2,\rho}$. The final equation (6.15e) represents the averaging operation performed in (6.14). The distributed nature of the method is highlighted in Algorithm 7.

Algorithm 7 Distributed Method of Multipliers

Initialize: $\mathbf{z}^{(0)} \in \mathbb{R}^{M_E}$
for t=0,..., **do**
  for all $i \in V$ **do** ▷ Primal and Dual Updates
    $\boldsymbol{\gamma}_{i,k}^{(t+1)} = \frac{1}{|\mathscr{N}_k(i)|} \sum_{j \in \mathscr{N}_k(i)} \mathbf{z}_{i|j,k}^{(t)}$
    $\mathbf{x}_i^{(t+1)} = \arg\min_{\mathbf{x}_i} \Big(f_i(\mathbf{x}_i) + \sum_{k \in \kappa} \Big(-\langle \mathbf{A}_{i,k}^T \boldsymbol{\gamma}_{i,k}^{(t+1)}, \mathbf{x}_i \rangle + \frac{\rho}{2|\mathscr{N}_k(i)|} \|\mathbf{A}_{i,k}\mathbf{x}_i - \mathbf{b}_{i,k}\|^2 \Big) \Big)$
    $\mathbf{w}_{i|j,k}^{(t+1)} = 2\boldsymbol{\gamma}_{i,k}^{(t+1)} - \mathbf{z}_{i|j,k}^{(t)} - 2\rho\left(\frac{\mathbf{A}_{i,k}\mathbf{x}_i}{|\mathscr{N}_k(i)|} - \frac{\mathbf{b}_{i,k}}{|\mathscr{N}_k(i)|}\right)$
  end for
  for all $k \in \kappa, i \in V_k, j \in \mathscr{N}_k(i)$ **do** ▷ Tx. Variables
    $\mathbf{Node}_j \leftarrow \mathbf{Node}_i(\mathbf{w}_{i|j,k}^{(t+1)})$
  end for
  for all $k \in \kappa, i \in V_k, j \in \mathscr{N}_k(i)$ **do** ▷ Aux. Updates
    $\mathbf{v}_{i|j,k}^{(t+1)} = \mathbf{w}_{j|i,k}^{(t+1)} - s_k \min\left\{\frac{\mathbf{w}_{i|j,k}^{(t+1)} + \mathbf{w}_{j|i,k}^{(t+1)}}{2}, \mathbf{0}\right\}$
    $\mathbf{z}_{i|j,k}^{(t+1)} = (1-\alpha^{(t)})\mathbf{z}_{i|j,k}^{(t)} + \alpha^{(t)}\mathbf{v}_{i|j,k}^{(t+1)}.$
  end for
end for

Here, we have made use of the fact that $\boldsymbol{\gamma}^{(t+1)}, \mathbf{w}^{(t+1)}, \mathbf{y}^{(t+1)}, \mathbf{z}^{(t)}$ all share the same structure as $\boldsymbol{\lambda}^{(t+1)}$, i.e. $\mathbf{z}^{(t)} = \left[\mathbf{z}_1^T, \cdots, \mathbf{z}_K^T\right]^T$ where $\mathbf{z}_k = [\mathbf{z}_{(1,2),k}^T, \cdots, \mathbf{z}_{(1,N),k}^T, \mathbf{z}_{(2,1),k}^T, \cdots, \mathbf{z}_{(N,N-1),k}^T]^T$

and so on for the other terms. Furthermore we have exploited the fact that $\boldsymbol{\gamma}^{(t+1)}, \boldsymbol{\lambda}^{(t+1)} \in \ker(\mathscr{L})$ such that $\boldsymbol{\gamma}^{(t+1)}_{i|j,k} = \boldsymbol{\gamma}^{(t+1)}_{i,k}$, $\boldsymbol{\lambda}^{(t+1)}_{i|j,k} = \boldsymbol{\lambda}^{(t+1)}_{i,k}$ $\forall k \in \kappa,\ i \in V_k,\ j \in \mathscr{N}_k(i)$.

6.4.4. CONVERGENCE GUARANTEES

Having formed a solver for general separable convex optimization problems, we now turn our attention to guaranteeing its convergence to an optimal solution. Thankfully, due to the use of a classic operator splitting approach in its derivation, this convergence follows directly from known results. Notably, for the considered class of objective functions, that is $f_i \in \Gamma_0\ \forall i \in V$ and that $\rho > 0$, $\sum_{t=0}^{+\infty} \alpha^{(t)} = +\infty$, the convergence of the proposed DMM algorithm follows directly from the convergence characteristics of the Krasnosel'skiĭ-Mann method (see [34, Theorem 5.15]).

In the case that $\forall t,\ \alpha^{(t)} = \alpha$, the auxiliary fixed point residual $\|\mathbf{z}^{(t+1)} - \mathbf{z}^{(t)}\|^2 \to 0$ at a rate of $\mathscr{O}\left(\frac{1}{t}\right)$ where $\bullet \to \bullet$ denotes convergence. As we are concerned with finite dimensional problems, it follows that $\exists \mathbf{z}^* \in \text{fix}\left(\mathbf{T}_{D,\rho}\right)$ so that $\mathbf{z}^{(t)} \to \mathbf{z}^*$ at the same rate.

6.4.5. DISTRIBUTED OPTIMIZATION OF GENERAL SEPARABLE PROBLEMS

While the prototype problem given in (6.1) may seem initially restrictive, in general any problem which exhibits both a separable objective and separable constraints can be solved by combining DMM with a primal lifting stage. Separability of the local functions f_i at each node can also be exploited to reduce the computational complexity of the primal updates.

Consider a general separable optimization problem given by

$$\begin{aligned} \min_{\mathbf{x}_i} \quad & \sum_{i \in V} \left(f_i(\mathbf{x}_i) + g_i(\mathbf{A}_{i,g}\mathbf{x}_i - \mathbf{b}_{i,g}) \right) \\ \text{s.t} \quad & \sum_{i \in V} h_i(\mathbf{x}_i) \leq \mathbf{0} \\ & \mathbf{A}_k \mathbf{x} - \mathbf{b}_k \underset{k}{\blacktriangleright} \mathbf{0}\ \forall k \in \kappa \end{aligned} \tag{6.16}$$

Here, the functions $g_i, h_i \in \Gamma_0\left(\mathbb{R}^{M_i}\right)\ \forall i \in V$.

The aim is to convert (6.16) to the form of (6.1), a point which can be achieved by introducing the additional primal variables $\mathbf{w}_i, \mathbf{z}_i$ at each node and the slack variables $\mathbf{y}_i$ such that (6.16) can be equivalently expressed as

$$\begin{aligned} \min_{\mathbf{w}_i, \mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i} \quad & \sum_{i \in V} \left(f_i(\mathbf{x}_i) + g_i(\mathbf{z}_i) \right) \\ \text{s.t} \quad & \sum_{i \in V} \mathbf{y}_i = \mathbf{0} \\ & \mathbf{A}_k \mathbf{x} - \mathbf{b}_k \underset{k}{\blacktriangleright} \mathbf{0}\ \forall k \in \kappa \\ & \mathbf{A}_{i,g}\mathbf{x}_i - \mathbf{b}_{i,g} = \mathbf{z}_i\ \forall i \in V \\ & h_i(\mathbf{w}_i) \leq \mathbf{y}_i,\ \mathbf{x}_i = \mathbf{w}_i\ \forall i \in V. \end{aligned} \tag{6.17}$$

The additional constraints enforce the equivalence of (6.16) and (6.17). Note that the only remaining constraints involving multiple nodes are affine with the convex constraints only acting locally at each node. By using indicator functions, we can shift these

non-affine inequality constraints to the objective so that (6.17) can be rephrased as

$$\begin{aligned} \min_{\mathbf{w}_i,\mathbf{x}_i,\mathbf{y}_i,\mathbf{z}_i} \quad & \sum_{i\in V}\left(f_i(\mathbf{x}_i)+g_i(\mathbf{z}_i)+\iota_{h_i(\mathbf{w}_i)\leq\mathbf{y}_i}\left(\mathbf{w}_i,\mathbf{y}_i\right)\right) \\ \text{s.t} \quad & \sum_{i\in V}\mathbf{y}_i=\mathbf{0} \\ & \mathbf{A}_k\mathbf{x}-\mathbf{b}_k \underset{k}{\blacktriangleright} \mathbf{0}\ \forall k\in\kappa \\ & \mathbf{x}_i-\mathbf{w}_i=\mathbf{0},\ \mathbf{A}_{i,g}\mathbf{x}_i-\mathbf{z}_i-\mathbf{b}_{i,g}=\mathbf{0}\ \forall i\in V. \end{aligned}$$

This is exactly in the form of (6.1) and thus can be directly solved via the DMM algorithm. In essence here, we have introduced a set of *virtual nodes* into the network with each handling a subset of the newly introduced primal variables. By separating the roles of f_i, g_i and $\delta_{h_i(\mathbf{w}_i)\leq\mathbf{y}_i}$ across these virtual nodes we can potentially reduce the complexity of the primal updates of the DMM algorithm. Furthermore as these virtual nodes only need to communicate with a single physical node, this approach also introduces no additional overhead in terms of communication cost making it an attractive choice for practical distributed implementations.

6.5. APPLICATION TO DISTRIBUTED SIGNAL PROCESSING

The role of the following section is to demonstrate how the proposed DMM algorithm can be used to solve a range of practical distributed optimization problems. For this purpose we have chosen three signal processing examples including a weighted beamforming application, the maximization of the channel capacity of a set of transmit antennas under a maximum power constraint and the optimization of a communal Markowitz portfolio.

6.5.1. RANDOM NETWORK MODELING

For all of the following examples, the networks we consider are generated via classic stochastic graph models. We consider three specific models: undirected Erdős-Rényi (ER) graphs [117], Watts-Strogatz (WS) small world graphs [118] and geometric random (GR) graphs [119]. Each different models, whilst being straightforward to implement, is constructed via different means and exhibit different network characteristics. In particular, WS and GR networks have been shown to be good candidates for modeling real world networks [118] [120]. Detailed explanations of the three topologies considered can be found in [117, 118, 119].

Each network was ensured to be sparsely connected and to form a single connected component as per our assumptions. This was achieved by configuring the associated parameters of each of the three network models. For the ER graphs, the probability of connection, which controls the set of constructed edges, was set to $\frac{\ln(N)}{N}$. This is referred to as the critical probability and generates networks with a low number of edges and a high probability the network being connected. For the WS graphs, the configuration process required two steps. Firstly, the initial K-hop lattice networks were configured so that $K=\lceil\ln(N)\rceil$. This choice generates networks with a similar number of edges to that of an equally sized ER graph. Secondly the probability of reconnection was configured to 5% to create only a limited number of random connections. Finally for the GR graphs,

a three dimensional unit cube was used to bound the locations of the randomly placed nodes while the transmission distance of the nodes was set to $r = \sqrt[3]{\frac{ln(N)}{N}}$ to again ensure a low number of edges and a high probability of connectivity.

6.5.2. A REFERENCE CENTRALIZED PR-SPLITTING METHOD

In addition to demonstrating the performance of the DMM algorithm in different network topologies, the following simulations also draw a comparison to a centralized PR-splitting based method. The motivation for this comparison is to offer insight into the degradation in performance experienced through the use of the proposed lifting. As with the DMM algorithm, the centralized implementation stems from the prototype problem in (6.1) which we can equivalently write in the unconstrained form

$$\min_{\mathbf{x}} f(\mathbf{x}) + \sum_{k\in\kappa} \iota_{\mathbf{A}_k\mathbf{x}-\mathbf{b}_k \underset{k}{\blacktriangleright} \mathbf{0}}(\mathbf{x}). \tag{6.18}$$

As with the extended dual problem, we can solve (6.18) by equivalently finding a solution of the monotonic inclusion

$$\mathbf{0} \in \partial f(\mathbf{x}) + \sum_{k\in\kappa} \partial \iota_{\mathbf{A}_k\mathbf{x}-\mathbf{b}_k \underset{k}{\blacktriangleright} \mathbf{0}}(\mathbf{x}).$$

By setting $\mathbf{T}_1 = \partial f$ and $\mathbf{T}_2 = \sum_{k\in\kappa} \partial \iota_{\mathbf{A}_k\mathbf{x}-\mathbf{b}_k \underset{k}{\blacktriangleright} \mathbf{0}}$, we can therefore apply averaged PR-splitting as in Section 6.3.9 to produce to the iterative centralized approach given in Algorithm 8. As with the DMM algorithm, the convergence of this approach follows from its relationship with Krasnosel'skiĭ-Mann type iterations. In all coming simulations, any reference to a centralized implementation refers to this approach.

Algorithm 8 Centralized PR-Splitting

1: **Initialize:** $\mathbf{z}^{(0)} \in \mathbb{R}^{M_V}$
2: **for** t=0,..., **do**
3: $\quad \mathbf{x}^{(t+1)} = \underset{\mathbf{x}}{\operatorname{argmin}} \left(f(\mathbf{x}) + \frac{\rho}{2} \|\mathbf{x} - \mathbf{z}^{(t)}\|^2 \right)$
4: $\quad \mathbf{y}^{(t+1)} = \underset{\mathbf{A}_k\mathbf{y}-\mathbf{b}_k \underset{k}{\blacktriangleright} \mathbf{0}\ \forall k\in\kappa}{\operatorname{argmin}} \left(\|\mathbf{y} - 2\mathbf{x}^{(t+1)} + \mathbf{z}^{(t)}\|^2 \right)$
5: $\quad z^{(t+1)} = (1-\alpha^{(t)})z^{(t)} + \alpha^{(t)} \left(2\mathbf{y}^{(t+1)} - 2\mathbf{x}^{(t+1)} + \mathbf{z}^{(t)} \right).$
6: **end for**

6.5.3. DISTRIBUTED BEAMFORMING

For our first application, consider the use of an N node wireless sensor network (WSN) where each node is equipped with a single receiver used to measure an acoustic signal. Given a set of noisy measurements taken by our network, the aim is to recover an unknown target signal of interest. The noise is assumed to be spatially uncorrelated Gaussian noise at each node with variance $\sigma_i^2 \ \forall\ i \in V$.

Such sources are typically processed in the time-frequency domain such that any delay can be expressed as a phase shift and thus as a complex scaling. For each frequency

bin our objective is to therefore to design a linear filter which preserves the signal in a target subspace $\mathbf{\Lambda}$ while reducing the power of received noise. Such a filter is a minimum variance distortionless response (MVDR) beamfomer for the specific case of uncorrelated noise [121] and can be computed as a solution to the following optimization problem:

$$\min_{\mathbf{w}} \sum_{i\in V} \frac{1}{2} x_i^H \sigma_i^2 x_i \quad \text{s.t.} \sum_{i\in V} \left(\mathbf{\Lambda}_i x_i - \frac{1}{N} \right) = 0,$$

Here, $\mathbf{x}$ denotes the vector of filter weights. Assuming that the elements of the vector $\mathbf{\Lambda}_i$ and σ_i are known locally at each node i, this problem is exactly in the form of (6.1).

Directly applying DMM, we can define the distributed MVDR beamformer given in Algorithm 9. Note that as there are no inequality constraints, the auxiliary updates have been simplified to remove the dependences on $\mathbf{S}$.

Algorithm 9 Distributed Beamforming for Uncorrelated Noise

Initialize: $\mathbf{z}^{(0)} \in \mathbb{R}^{M_E}$
for t=0,..., **do**
  for all $i \in V$ **do** ▷ Primal and Dual Updates
    $\gamma_i^{(t+1)} = \frac{1}{|\mathcal{N}(i)|} \sum_{j\in\mathcal{N}(i)} z_{i|j}^{(t)}$
    $x_i^{(t+1)} = \left(\sigma_i^2 + \frac{\rho \mathbf{\Lambda}_i^H \mathbf{\Lambda}_i}{|\mathcal{N}(i)|} \right)^{-1} \left(\mathbf{\Lambda}_i^H \gamma_i^{(t+1)} + \frac{\rho \mathbf{\Lambda}_i^H}{N|\mathcal{N}(i)|} \right)$
    $w_{i|j}^{(t+1)} = 2\gamma_i^{(t+1)} - z_{i|j}^{(t)} - 2\rho \left(\frac{x_i^{(t+1)}}{|\mathcal{N}(i)|} - \frac{1}{N|\mathcal{N}(i)|} \right)$
  end for
  for all $i \in V, j \in \mathcal{N}(i)$ **do** ▷ Transmit Variables
    $\mathbf{Node}_j \leftarrow \mathbf{Node}_i(w_{i|j}^{(t+1)})$
  end for
  for all $i \in V, j \in \mathcal{N}(i)$ **do** ▷ Auxiliary Variables
    $z_{i|j}^{(t+1)} = (1-\alpha^{(t)}) z_{i|j}^{(t)} + \alpha^{(t)} w_{j|i}^{(t+1)}$.
  end for
end for

To demonstrate the performance of the proposed method, three 1000 node networks were generated as per the different methods outlined in Section 6.5.1. The noise variances σ_i and target subspace $\mathbf{\Lambda}_i$ which were used for all three networks were generated randomly. The step size of DMM was empirically chosen for each network to optimize convergence rate. Additionally, $\forall k \in \mathbb{N}, \alpha^{(k)} = \frac{1}{2}$ was selected, resulting in a DR splitting variant of the DMM algorithm. The primal convergence of the algorithm is given in Figure 6.7 in addition to the convergence of the objective function and relative objective error $\|f(\mathbf{x}) - f(\mathbf{x}^*)\|^2$. In addition to the three types of networks considered, we also include results for the reference centralized approach where again the step size parameter ρ was empirically optimized. The final precision of both algorithms is due to the physical hardware utilized.

For the given number of nodes, the ability of the algorithm to achieve machine precision in less than N iterations is a satisfying result. Furthermore, due to the quadratic nature of the local optimization problems at each node, the primal updates are analytic and inexpensive to compute. The convergence rate is also far better than the asymptotic bound of Krasnosel'skiĭ-Mann type schemes [34, Theorem 5.15] which most likely stems from the strong convexity and smoothness of each local objective function. Of additional interest is the fact that the proposed DMM offers comparable performance to that of the centralized PR-splitting method. This is most clearly demonstrated in the primal variables with the primal error only converging twice as fast in the centralized case. In this way, the additional dual lifting has not come at a considerable reduction in convergence rate while allowing for a fully distributable implementation.

6.5.4. GAUSSIAN CHANNEL CAPACITY MAXIMIZATION

As a second example, consider a WSN of N independent antennas trying to communicate a signal back to a target location over a set of N additive white Gaussian channels (AWGNs). Given a local bandwidth B_i for each channel, the objective of this problem is to optimally configure the transmission power of the antennas ($\mathbf{x}$) to maximize channel capacity under a total power constraint. From the Shannon-Hartley theorem [122], the capacity of each channel (C_i) is given by

$$C_i = B_i \log_2\left(1 + \frac{x_i}{\sigma_i}\right) = \frac{B_i(\ln(\sigma_i + x_i) - \ln(\sigma_i))}{\ln(2)},$$

where σ_i^2 is the noise variance of the ith channel.

The channel capacity maximization problem under a maximum power constraint can be rephrased as a convex optimization problem of the form

$$\min_{\mathbf{x}} -\sum_{i\in V} B_i \ln(x_i+\sigma_i) \text{ s.t } \sum_{i\in V} x_i = 1, \mathbf{x} \geq \mathbf{0},$$

where the non-negativity constraints stem from the fact that power is non-negative. If assuming each node i has an additional local maximum transmission power constraint $x_i \leq \beta_i$, the final optimization problem is given by

$$\min_{\mathbf{x}} -\sum_{i\in V} B_i \ln(x_i+\sigma_i) \text{ s.t } \sum_{i\in V} x_i = 1, \boldsymbol{\beta} \geq \mathbf{x} \geq \mathbf{0}, \tag{6.19}$$

where the vector $\boldsymbol{\beta}$ is the stacked vector of all β_i.

By using indicator functions to move the local constraints to the objective, (6.19) can be converted into the form of (6.1). The resulting problem is given by

$$\begin{aligned} \min_{\mathbf{x}} \quad & \sum_{i\in V} \left(-B_i \ln(x_i+\sigma_i) + \iota_{\geq 0}(x_i) + \iota_{\leq \beta_i}(x_i) \right) \\ \text{s.t} \quad & \sum_{i\in V} \left(x_i - \frac{1}{N} \right) = 0, \end{aligned}$$

after which we can directly apply the proposed DMM algorithm. This is summarized in Algorithm 10.

Algorithm 10 Distributed Channel Capacity Maximization

Initialize: $\mathbf{z}^{(0)} \in \mathbb{R}^{M_E}$
for t=0,..., **do**
  for all $i \in V$ **do** ▷ Primal and Dual Updates
    $\gamma_i^{(t+1)} = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} z_{i|j}^{(t)}$
    $x_i^{(t+1)} = \underset{0 \le x_i \le \beta_i}{\arg\min} \Big(-B_i \ln(x_i + \sigma_i) - \langle \gamma_i^{(t+1)}, x_i \rangle + \frac{\rho}{2|\mathcal{N}(i)|} ||x_i - \frac{1}{N}||^2 \Big)$
    $w_{i|j}^{(t+1)} = 2\gamma_i^{(t+1)} - z_{i|j}^{(t)} - 2\rho \left(\frac{x_i^{(t+1)}}{|\mathcal{N}(i)|} - \frac{1}{N|\mathcal{N}(i)|} \right)$
  end for
  for all $i \in V, j \in \mathcal{N}(i)$ **do** ▷ Transmit Variables
    $\mathbf{Node}_j \leftarrow \mathbf{Node}_i(w_{i|j}^{(t+1)})$
  end for
  for all $i \in V, j \in \mathcal{N}(i)$ **do** ▷ Auxiliary Variables
    $z_{i|j}^{(t+1)} = (1 - \alpha^{(t)}) z_{i|j}^{(t)} + \alpha^{(t)} w_{j|i}^{(t+1)}$.
  end for
end for

Again, we consider the performance of this algorithm in three random networks, each comprised of 100 nodes. For each, the same randomly generated σ_i and β_i were used. The resulting convergence characteristics are given in Figure 6.8 where the step size ρ was chosen to optimize the convergence rate for each network. It was found that for this particular problem large step sizes (on the order of 10^3) provided much faster convergence rates. As with the DS beamformer, $\forall k \in \mathbb{N}, \alpha^{(k)} = \frac{1}{2}$ was selected. Again, the finite noise floor observed is due to the finite numerical precision of the simulations.

We can observe that the algorithm quickly converges, reaching a primal precision of 10^{-15} in 200-350 iterations. As with the distributed DS beam-former, the proposed algorithm exhibits a linear convergence rate. As in the case of the previous distributed beamforming example, this most likely stems from the strong convexity and smoothness of each local problem over the allowed domain $\beta_i \ge x_i \ge 0 \; \forall i \in V$ although no proof of this is offered at this time. Finally, we can also observe that, when compared with the centralized approach, there is not a significant degradation in convergence rate through the use of the DMM algorithm. In particular, the convergence of the primal iterates to their optimal state takes roughly twice as long as the centralized solution.

6.5.5. PORTFOLIO OPTIMIZATION

As a final example we consider the task of Markowitz portfolio optimization [123]. While this problem in its standard form is inherently non-distributed, here we consider a variant of this problem for a collaborative network of investors. In the non-collaborative case [48, Sec. 4.4.1], the basic premise of this problem is that each node within the network has a local portfolio of stocks or assets into which they want to invest a given local wealth w_i while minimizing the risk of the investment for a certain return r_i. The return on the

set of such stocks is modeled by the random vector $\mathbf{p}_i \in \mathbb{R}^{M_i}$ such that $\mathbb{E}\left[\mathbf{p}_i\right] = \bar{\mathbf{p}}_i \in \mathbb{R}^{M_i}$ and $\mathbb{E}\left[(\mathbf{p}_i - \bar{\mathbf{p}}_i)^2\right] = \mathbf{Q}_i \in \mathbb{R}^{M_i \times M_i}$. The risk of investment can therefore be modeled as a quadratic cost. Ultimately, each node $i \in V$ wants to solve a problem of the form

$$\min_{\mathbf{x}_i} \frac{1}{2}\mathbf{x}_i^T \mathbf{Q}_i \mathbf{x}_i \text{ s.t. } \bar{\mathbf{p}}_i^T \mathbf{x}_i \geq r_i, \mathbf{1}^T \mathbf{x}_i = w_i, \mathbf{x}_i \geq 0,$$

where $\mathbf{x}_i$ denotes the vector of investments made in the stocks considered and the final set of constraints indicate that we do not want to considering short positions in our investments.

We consider a variant of this problem where the set of nodes work together to lower the total risk of the network by investing in each others portfolios while maintaining their own individual return ambitions. Additionally we incorporate local investment constraints which require a certain amount of each nodes wealth to be invested locally. This reflects a prior favoritism by an investor in their own work. The collaborative variant of the portfolio optimization problem is given by

$$\begin{aligned} \min_{\mathbf{x}} \quad & \sum_{i \in V} \frac{1}{2}\mathbf{x}_i^T \mathbf{Q}_i \mathbf{x}_i \\ \text{s.t.} \quad & \sum_{i \in V} \left(\bar{\mathbf{p}}_i^T \mathbf{x}_i - r_i\right) \geq 0, \sum_{i \in V} \left(\mathbf{1}^T \mathbf{x}_i - w_i\right) = 0 \\ & \mathbf{x}_i \geq 0, \mathbf{1}^T \mathbf{x}_i \geq \eta_i w_i \; \forall \; i \in V. \end{aligned} \tag{6.20}$$

Here, the first two constraints ensure that the total return and total wealth requirements of the network are satisfied but in this case in a collaborative rather than node based case. The third constraint is the non-shorting constraint while the final constraint captures the required local portfolio investment at each node. The variables $\eta_i \in [0,1]$ capture the required ratio of local portfolio investment to local wealth.

Defining the matrices $\mathbf{A}_i$ and vector $\mathbf{b}_i$ as

$$\mathbf{A}_i = \begin{bmatrix} \bar{\mathbf{p}}_i^T & \mathbf{0} \\ \mathbf{0} & \mathbf{1}^T \end{bmatrix}, \mathbf{b}_i = \begin{bmatrix} r_i \\ w_i \end{bmatrix} \; \forall i \in V,$$

and by utilizing indicator functions to shift the local constraints to the objective function, (6.20) can be rewritten as

$$\begin{aligned} \min_{\mathbf{x}} \quad & \sum_{i \in V} \left(\frac{1}{2}\mathbf{x}_i^T \mathbf{Q}_i \mathbf{x}_i + \iota_{\mathbf{x}_i \geq \mathbf{0}}(\mathbf{x}_i) + \iota_{\mathbf{1}^T \mathbf{x}_i \geq \eta_i w_i}(\mathbf{x}_i) \right) \\ \text{s.t} \quad & \sum_{i \in V} (\mathbf{A}_i \mathbf{x}_i - \mathbf{b}_i) \blacktriangleright 0, \end{aligned} \tag{6.21}$$

It follows that (6.21) is exactly in the form given by (6.1). Therefore, applying the DMM algorithm, (6.20) can be solved distributedly via Algorithm 11. In Algorithm 11, $\odot$ is used to denote the element-wise or Hadamard product of two vectors.

For demonstration purposes we consider an Erdős-Rényi network comprised of 100 nodes, each with a local portfolio size of 20 elements. The system parameters $(\mathbf{Q}_i, \bar{\mathbf{p}}_i, r_i, w_i, \eta_i)$

Algorithm 11 Collaborative Markowitz Portfolio Optimization

Initialize: $\mathbf{z}^{(0)} \in \mathbb{R}^{M_E}$
for t=0,..., **do**
  for all $i \in V$ **do** ▷ Primal and Dual Updates
    $\boldsymbol{\gamma}_i^{(t+1)} = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \mathbf{z}_{i|j}^{(t)}$
    $\mathbf{x}_i^{(t+1)} = \underset{\mathbf{x}_i \geq 0, \mathbf{1}^T \mathbf{x}_i \geq \eta_i w_i}{\arg\min} \left(\frac{1}{2} \mathbf{x}_i^T \mathbf{Q}_i \mathbf{x}_i - \langle \mathbf{A}_i \boldsymbol{\gamma}_i^{(t+1)}, \mathbf{x}_i \rangle + \frac{\rho}{2|\mathcal{N}(i)|} \|\mathbf{A}_i \mathbf{x}_i - \mathbf{b}_i\|^2 \right)$
    $\mathbf{w}_{i|j}^{(t+1)} = 2\boldsymbol{\gamma}_i^{(t+1)} - \mathbf{z}_{i|j}^{(t)} - 2\rho \left(\frac{\mathbf{A}_i \mathbf{x}_i^{(t+1)}}{|\mathcal{N}(i)|} - \frac{\mathbf{b}_i}{|\mathcal{N}(i)|} \right)$
  end for
  for all $i \in V, j \in \mathcal{N}(i)$ **do** ▷ Transmit Variables
    $\mathbf{Node}_j \leftarrow \mathbf{Node}_i(\mathbf{w}_{i|j}^{(t+1)})$
  end for
  for all $i \in V, j \in \mathcal{N}(i)$ **do** ▷ Auxiliary Variables
    $\mathbf{v}_{i|j}^{(t+1)} = \mathbf{w}_{j|i}^{(t+1)} - \begin{bmatrix} 1 \\ 0 \end{bmatrix} \odot \min \left\{ \mathbf{w}_{i|j}^{(t+1)} + \mathbf{w}_{j|i}^{(t+1)}, \mathbf{0} \right\}$
    $\mathbf{z}_{i|j}^{(t+1)} = (1 - \alpha^{(t)}) \mathbf{z}_{i|j}^{(t)} + \alpha^{(t)} \mathbf{v}_{i|j}^{(t+1)}.$
  end for
end for

were generated randomly and ρ was empirically selected to optimize convergence rate. The resulting convergence characteristics are included in Figure 6.9.

As with our other two examples, we can observe the familiar linear convergence rate of the primal error in addition to the rapid optimality of the configuration. In particular, within around 200 iterations the total portfolio risk is within 0.1% of the optimal configuration. In contrast to the other two examples however, here we can note a significant degradation in convergence between the DMM implementation and the centralized approach. Additionally, the DMM algorithm required far more iterations to converge in this instance than for the other examples despite having a comparable network size. While not as encouraging for the usability of DMM as the other two examples, this result is more inline with our expectation for a distributed method.

6.6. CONCLUSIONS

In this paper we have presented a novel method for distributed optimization of separable convex problems. In contrast to other existing methods within the literature, the prototype problem of the proposed DMM algorithm is not restricted based on the topology of the underlying network. This allows DMM to be used in a much broader range of applications whilst preserving its distributed operation. Furthermore, the derivation for this method is based on classical monotone operator theory with the DMM algorithm itself being based on a combination of Peaceman-Rachford splitting and Krasnosel'skiĭ-Mann iterations, providing an intuitive interpretation of the approach. The

convergence of DMM follows from the existing results for these methods. The use of DMM was demonstrated for a range of practical signal processing problems including beamforming, channel capacity maximization and portfolio optimization for a range of network types. Overall, DMM demonstrates that any separable problem can be solved in a distributed manner in undirected networks and thus provides a novel tool for distributed convex optimization.

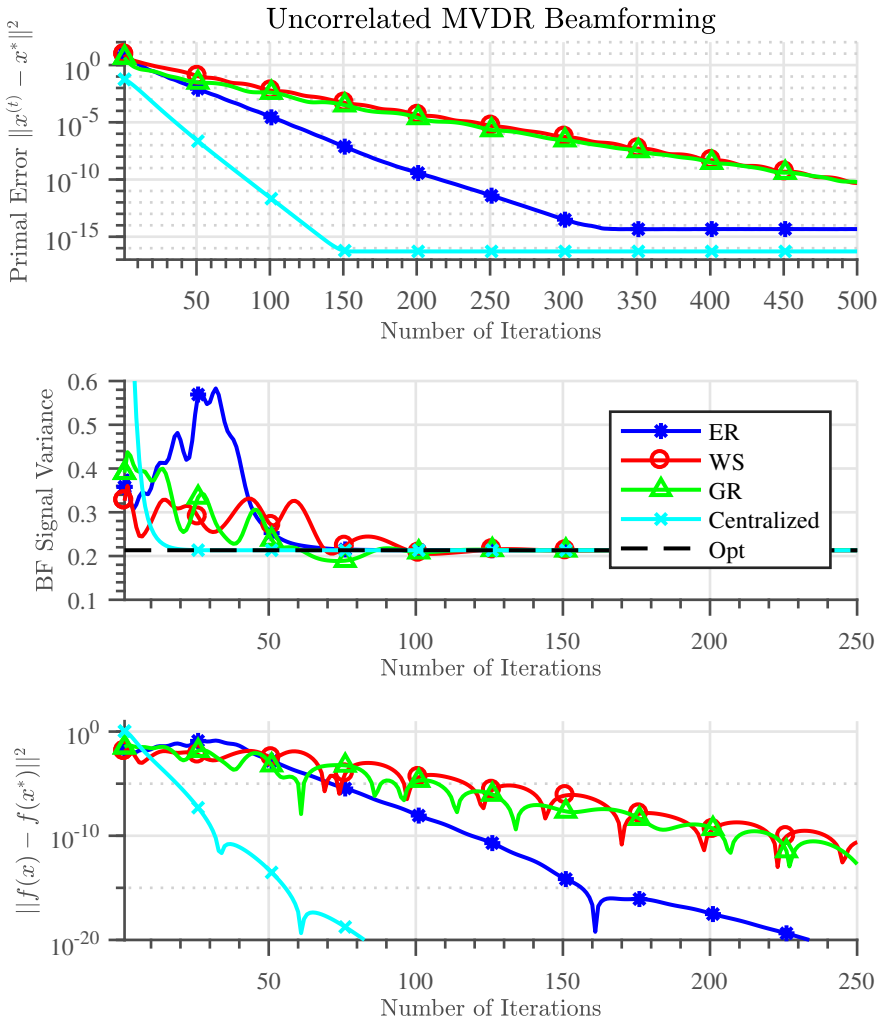


Figure 6.7: An example of uncorrelated MVDR beamforming for a 1000 node network via the DMM algorithm. We compare the primal mean squared error, the variance of the resulting beamformed signal and the relative objective error for the three networks in question and a centralized PR-splitting based approach.

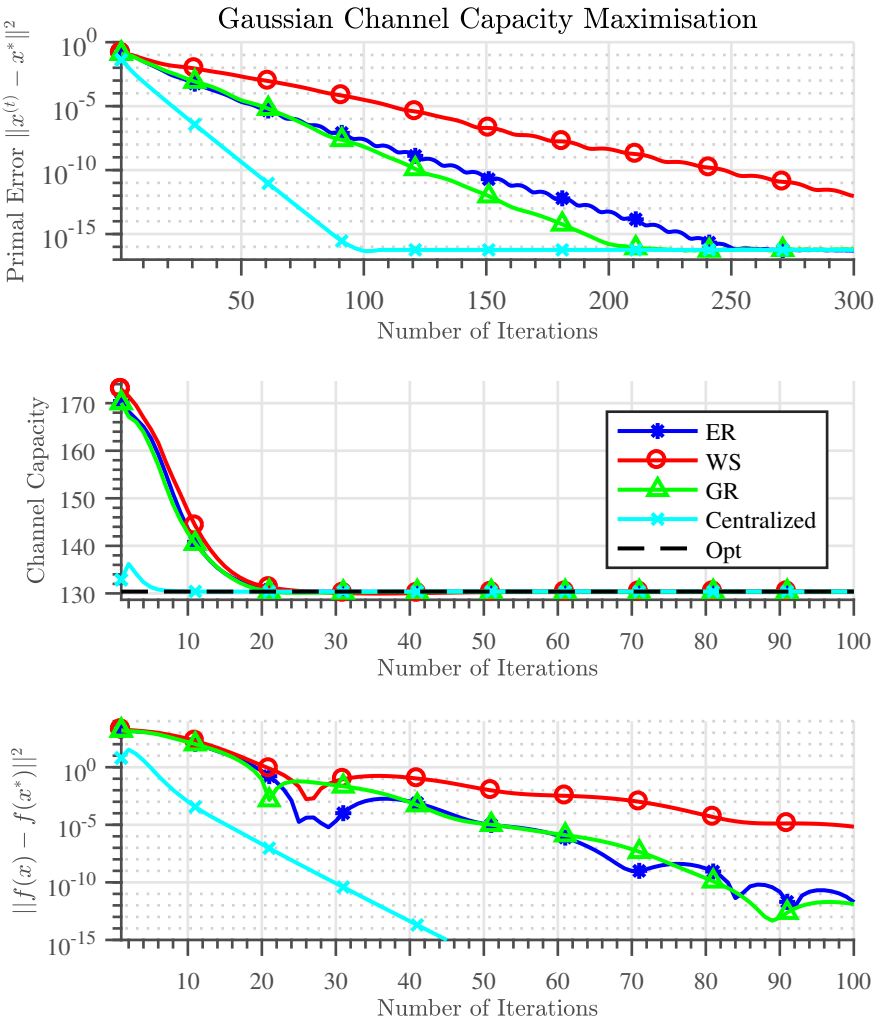


Figure 6.8: An example of channel capacity maximization for a 100 node network via the DMM algorithm. Note that the initial overshoot in terms of channel capacity stems from the violation of the constraint functions for these iterations

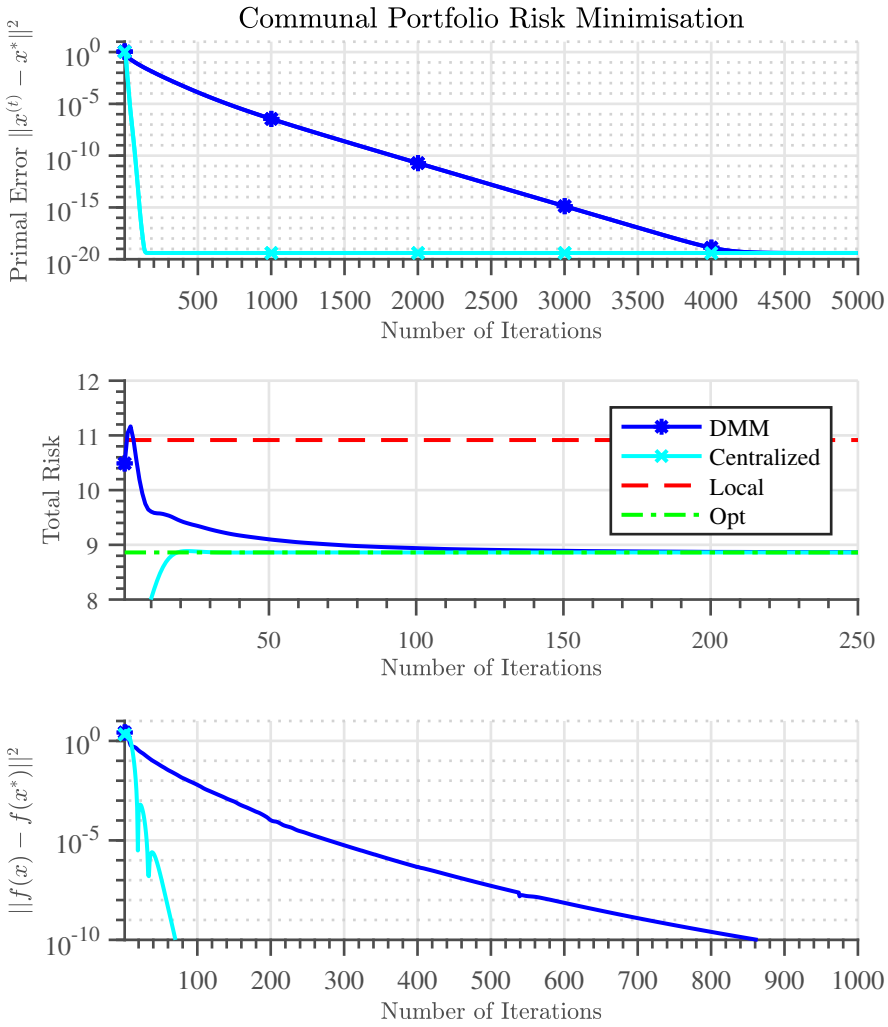


Figure 6.9: Distributed Markowitz portfolio optimization for a 100 node network with 20 stocks per node solved via the DMM algorithm. The red line denotes the total risk of the network without collaboration, the green is the optimal configuration.

APPENDICES

6.A. PROOF OF LEMMA 6.4.1

From (6.13), the resolvent of $\mathbf{z}^{(t)}$ is defined as

$$\boldsymbol{\lambda}^{(t+1)} = \mathbf{J}_{\mathbf{T}_1,\rho}\left(\mathbf{z}^{(t)}\right).$$

From the definition of $\mathbf{T}_1$ (6.11a), it follows that

$$\boldsymbol{\lambda}^{(t+1)} \in \mathbf{z}^{(t)} - \rho\left(\mathbf{C}\partial f^*\left(\mathbf{C}^T\boldsymbol{\lambda}^{(t+1)}\right) - \mathbf{d} + \partial \iota_{\mathscr{C}_2}\left(\boldsymbol{\lambda}^{(t+1)}\right)\right).$$

Defining, $\mathbf{x} \in \partial f^*\left(\mathbf{C}^T\boldsymbol{\lambda}\right)$, for a given $\mathbf{x}^{(t+1)}$, $\boldsymbol{\lambda}^{(t+1)}$ can be computed as

$$\boldsymbol{\lambda}^{(t+1)} = \underset{\mathscr{L}\boldsymbol{\lambda}=\mathbf{0}}{\arg\min}\left(\frac{1}{2}\|\boldsymbol{\lambda} - \mathbf{z}^{(t)} + \rho\left(\mathbf{C}\mathbf{x}^{(t+1)} - \mathbf{d}\right)\|^2\right) \tag{6.22}$$

Given the structure of $\mathbf{C}$ and $\mathbf{d}$, $\mathscr{L}\left(\mathbf{C}\mathbf{x}^{(t+1)} - \mathbf{d}\right) = \mathbf{0}$ by design. We can therefore perform a change of variables in the dual update of (6.22) by defining the additional variable

$$\boldsymbol{\gamma}^{(t+1)} = \boldsymbol{\lambda}^{(t+1)} + \rho\left(\mathbf{C}\mathbf{x}^{(t+1)} - \mathbf{d}\right).$$

From (6.22), $\boldsymbol{\gamma}^{(t+1)}$ can be computed as

$$\boldsymbol{\gamma}^{(t+1)} = \underset{\mathscr{L}\boldsymbol{\gamma}=\mathbf{0}}{\arg\min}\left(\frac{1}{2}\|\boldsymbol{\gamma} - \mathbf{z}^{(t)}\|^2\right) = \underset{\ker(\mathscr{L})}{\Pi}\mathbf{z}^{(t)}.$$

Using the definitions of $\mathbf{x}^{(t+1)}$ and $\boldsymbol{\gamma}^{(t+1)}$, it follows that

$$\mathbf{x}^{(t+1)} \in \partial f^*\left(\mathbf{C}^T\left(\boldsymbol{\gamma}^{(t+1)} - \rho\left(\mathbf{C}\mathbf{x}^{(t+1)} - \mathbf{d}\right)\right)\right),$$

and thus that the primal updates satisfy the inclusion

$$\partial f\left(\mathbf{x}^{(t+1)}\right) \ni \mathbf{C}^T\left(\boldsymbol{\gamma}^{(t+1)} - \rho\left(\mathbf{C}\mathbf{x}^{(t+1)} - \mathbf{d}\right)\right).$$

The primal updates $\mathbf{x}^{(t+1)}$ can therefore be computed as

$$\mathbf{x}^{(t+1)} = \underset{\mathbf{x}}{\arg\min}\left(f(\mathbf{x}) + \frac{\rho}{2}\|\mathbf{C}\mathbf{x} - \mathbf{d}\|^2 - \langle \mathbf{C}^T\boldsymbol{\gamma}^{(t+1)}, \mathbf{x}\rangle\right).$$

The definition of the reflected resolvent completes the proof. □

6.B. PROOF OF LEMMA 6.4.2

Define the output of the resolvent operator for a given iterate by the vector

$$\mathbf{y}^{(t+1)} = \mathbf{J}_{\rho,\mathbf{T}_2}\left(\mathbf{w}^{(t+1)}\right).$$

By the definition of the operator $\mathbf{T}_2$ (6.11b), $\mathbf{y}^{(k+1)}$ can be computed as the minimizer of

$$\begin{aligned} \min_{\mathbf{y}} \quad & \frac{1}{2}\|\mathbf{y}-\mathbf{w}^{(t+1)}\|^2 \\ \text{s.t.} \quad & (\mathbf{I}-\mathbf{P})\,\mathbf{y} = \mathbf{0},\ \mathbf{S}\mathbf{y}^* \geq \mathbf{0}. \end{aligned} \tag{6.23}$$

Equivalently, it can be computed by solving the KKT system of equations [48] given by

$$\begin{aligned} \mathbf{y}^* - \mathbf{w}^{(t+1)} - (\mathbf{I}-\mathbf{P})\,\boldsymbol{\mu}^* - \mathbf{S}\boldsymbol{\nu}^* = \mathbf{0}, \\ (\mathbf{I}-\mathbf{P})\,\mathbf{y}^* = \mathbf{0},\ \mathbf{S}\mathbf{y}^* \geq \mathbf{0},\ \boldsymbol{\nu}^* \geq \mathbf{0},\ \left(\mathbf{S}\mathbf{y}^*\right) \odot \boldsymbol{\nu}^* = \mathbf{0}, \end{aligned}$$

where $\odot$ denotes the Hadamard product, the element-wise product of two vectors. The variables $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ denote the dual variables associated with the constraints in (6.23).

Combining the first two equalities and using the self-inverse property of $\mathbf{P}$, $\mathbf{P}^2 = \mathbf{I}$, it follows that

$$-(\mathbf{I}-\mathbf{P})\,\boldsymbol{\mu}^* = \frac{1}{2}(\mathbf{I}-\mathbf{P})\left(\mathbf{w}^{(t+1)} + \mathbf{S}\boldsymbol{\nu}^*\right),$$

and thus, that

$$\mathbf{y}^* = \frac{1}{2}(\mathbf{I}+\mathbf{P})\left(\mathbf{w}^{(t+1)} + \mathbf{S}\boldsymbol{\nu}^*\right).$$

For those dual variables corresponding to equality constraints, the corresponding entries of $\boldsymbol{\nu}$ variables play no role. For the remaining dual variables, combining the non-negativity of $\mathbf{y}^*$ and $\boldsymbol{\nu}^*$ with the complementary slackness condition $\left(\mathbf{S}\mathbf{y}^*\right) \odot \boldsymbol{\nu}^* = \mathbf{0}$, it follows that $\mathbf{y}^* = \max\{\frac{1}{2}(\mathbf{I}+\mathbf{P})\,\mathbf{w}^{(t+1)}, \mathbf{0}\}$ where $\max\{\bullet,\bullet\}$ is an abuse of notation used to denote element-wise maximization between two vectors. The vector $\mathbf{y}^{(t+1)}$ can therefore be succinctly computed as

$$\begin{aligned} \mathbf{y}^{(t+\frac{1}{2})} &= \frac{1}{2}(\mathbf{I}+\mathbf{P})\,\mathbf{w}^{(t+1)}, \\ \mathbf{y}^{(t+1)} &= \mathbf{y}^{(t+\frac{1}{2})} - \mathbf{S}\min\left\{\mathbf{y}^{(t+\frac{1}{2})}, \mathbf{0}\right\}, \end{aligned}$$

where, similar to $\max\{\bullet,\bullet\}$, $\min\{\bullet,\bullet\}$ denotes element-wise minimization. The definition of the reflected resolvent completes the proof. □

7

DISTRIBUTED CONSENSUS OVER TIME VARYING NETWORKS

Thomas Sherson

"Music has no effect on research work, but both are born of the same source and complement each other through the satisfaction they bestow"

Albert Einstein

In this chapter we demonstrate a novel method for performing distributed consensus in time-varying undirected networks. The proposed method combines classical monotone operator splitting approaches with a specific time-varying metric choice to produce an algorithm which allows the network topology to vary at each iteration. Based on averaged Peaceman-Rachford splitting, we demonstrate guaranteed convergence in solving strongly convex problems as well as highlighting how in practice more general closed, convex and proper functions can be solved via this method as well. Additionally, the proposed algorithm provides insight into how the topology of a given network manifests itself in the implementation of distributed consensus through the formation of a weighted graph Laplacian operator.

7.1. INTRODUCTION

The omnipotence of wireless equipped devices in modern society is revolutionizing many of the ways in which we conduct ourselves. From the emergence of new paradigms such as the "Internet of Things" (IoT) [5] and blockchain technologies [91] through to more salient tasks such as improving power generation in smart grids [67] and assisting in the navigation of autonomous fleets of cars [4], these devices are playing an ever increasing role in many of the fundamental utilities of our world. Motivated by this point, the last few decades have seen a significant growth in the interest in parallel and distributed methods of computation. By leveraging the computational capabilities of all devices within a network without the need for centralized data aggregation, these methods promise to provide adaptive and robust solutions to many signal processing tasks. Results already exist in many areas such as cloud computing where problem parallelization is utilized to distribute computational tasks across multiple machines. Similarly, a range of algorithms from distributed averaging [12, 72, 92, 93], to graph filtering [17, 73, 74, 95], belief propagation [13, 14, 15, 94] and convex optimization [18, 19, 79, 96, 97] have been proposed within the literature to facilitate more general signal processing in such contexts.

One of the main drawbacks of many existing distributed approaches lies in the assumption that the underlying network is time invariant. While in some cases, these networks may indeed be static, more generally we expect network topologies to change over time. Such time varying features may stem from a variety of sources including, but not limited to, link failure between nodes (e.g., loss of a powerline in a smart grid), node movement within the network (e.g., autonomous cars moving as part of a fleet), variations in transmission power of nodes (e.g., cellphones switching to a low power mode when their batteries are almost drained) and more. The design of distributed algorithms to naturally handle these changes is therefore an attractive and, in many cases, necessary requirement for applying such methods to real world problems.

While there are methods available within the literature to perform distributed optimization in a time varying context [11, 40, 60, 124], in general the types of problems which can be solved limits their applicability to real world problems. The purpose of this chapter is to develop a distributed solver for use in time varying networks based on monotone operator theory with the desire to broaden the class of tractable problems via such methods. Notably we introduce the Time Varying Distributed Consensus method and verify the sufficiency of strong convexity to guarantee its convergence. We empirically show that for more general closed, convex and proper functions convergence can also be achieved and as a biproduct, postulate the existence of a more general convergence proof.

7.1.1. RELATED WORK

The work in this chapter builds upon that of many key figures within the literature including Rockafellar, whose fundamental work on network optimization [76] and the relation between convex optimization and monotone operator theory [26, 27, 28] remains central to many results to this day. Additional works by the likes of Bertsekas and Tsitsiklis [29, 30, 31] are similarly important to the narrative where again separability was used as a mechanism to design a range of distributed algorithms.

Given the emergence of large scale networking problems in recent years, these approaches have been undergoing a revival within the literature [29, 30, 31, 32, 33]. As previously mentioned however, many of these approaches are designed for time invariant network topologies and do not easily generalize to the time varying case. While a few methods offer the ability to adapt with the network, such as randomized gossip [12], such approaches are often limited to linear signal processing tasks. In contrast, distributed optimization type methods are traditionally able to address a much larger range of nonlinear tasks but are also typically based on time invariant network models.

A notable exception of the rule of time invariant network models are the subgradient type algorithms for distributed optimization [11]. One of the earliest such methods was proposed in [10], and follows as a natural extension of subgradient descent applied to centralized network topologies. While in practice only approximating the standard subgradient type method, the literature for these approaches has matured significantly over the last ten years with variants that provide guaranteed convergence for asynchronous operation where nodes do not necessarily update at the same time [40], directed network topologies [11, 60] and more. Additionally a large amount of effort has gone into understanding the effects of other practical considerations such as the impact of network topology [102], link failure [61] and even quantization [125] on such algorithms.

Unfortunately, in terms of convergence, the approaches mentioned above typically require strict assumptions on the class of functions considered, namely that the functions have Lipschitz continuous gradients and in practice may prove unnecessarily restrictive for practical use. The motivation for this chapter was whether an algorithm could be constructed for performing distributed consensus in time varying networks using the perspective of monotone operator theory. In particular, we wanted to see if guaranteed convergence could be demonstrated for more general functional classes, hence broadening the number of distributed problems which could be addressed in a time varying context.

7.1.2. MAIN CONTRIBUTIONS

The main contributions of this chapter are two fold. Firstly, by designing a solver for distributed consensus based on averaged Peaceman-Rachford splitting, we introduce an equivalent algorithm tailored to the task of performing distributed consensus in time varying networks via the use of a time varying change of variables at each iteration. Secondly, we demonstrate guaranteed convergence for closed, convex and proper functions in the case of time invariant networks and for strongly convex functions demonstrate a convergence proof for time varying networks as well. Furthermore, we provide empirical evidence which suggests that a more general convergence proof may exist for closed, convex and proper functions in the time varying case.

7.1.3. ORGANIZATION OF CHAPTER

The remainder of this chapter is organized as follows. Section 7.2 includes the nomenclature used throughout the text. Section 7.3 provides the derivation for the template distributed consensus algorithm based on monotone operator theory. Section 7.4 demonstrates a compact interpretation of the algorithm to remove dependence on the network topology and demonstrates convergence in the static network case. Section 7.6 expands

the proposed method to time-varying networks and demonstrates guaranteed convergence of the fixed point residual for strongly convex functions. Section 7.7 demonstrates the use of the algorithm in solving two simple distributed optimization problems. Finally Section 7.8 presents our conclusions.

7.2. NOMENCLATURE

In this work we denote by $\mathbb{R}$ the set of real numbers, by $\mathbb{R}^N$ the set of real column vectors of length N and by $\mathbb{R}^{M\times N}$ the set of M by N real matrices. Let $\mathscr{X},\mathscr{Y}\subseteq\mathbb{R}^N$. A set valued operator $\mathbf{T}:\mathscr{X}\rightarrow\mathscr{Y}$ is defined by its graph, $\text{gra}(\mathbf{T})=\{(\mathbf{x},\mathbf{y})\in\mathscr{X}\times\mathscr{Y}\mid\mathbf{y}\in\mathbf{T}(\mathbf{x})\}$. Similarly, the notion of an inverse of an operator $\mathbf{T}^{-1}$ is defined via its graph so that $\text{gra}\left(\mathbf{T}^{-1}\right)=\{(\mathbf{y},\mathbf{x})\in\mathscr{Y}\times\mathscr{X}\mid\mathbf{y}\in\mathbf{T}(\mathbf{x})\}$. $\mathbf{J}_{\mathbf{T},\rho}=\left(\mathbf{I}+\rho\mathbf{T}\right)^{-1}$ denotes the resolvent of an operator while $\mathbf{R}_{\mathbf{T},\rho}=2\mathbf{J}_{\mathbf{T},\rho}-\mathbf{I}$ denotes the reflected resolvent (Cayley operator). The fixed-point set of $\mathbf{T}$ is denoted by $\text{fix}(\mathbf{T})=\{\mathbf{x}\in\mathscr{X}\mid\mathbf{T}(\mathbf{x})=\mathbf{x}\}$.

7.3. DISTRIBUTED CONSENSUS

In this section we introduce a basic distributed consensus problem which forms the central focus of this chapter and demonstrate the derivation of a distributed solver based on averaged Peaceman-Rachford splitting. This derivation follows along the same lines as our previous chapters but incorporates an additional change of variables which depends on the underlying network topology. In the later portion of this chapter, we will see that this algorithm also forms the basis of a novel method for consensus in time varying networks (see Section 7.6).

7.3.1. PROBLEM DEFINITION

Consider a simple undirected network consisting of N nodes with which we want to perform convex optimization in a distributed manner. The associated graphical model of such a network is given by $G(V,E)$ where $V=\{1,\cdots,N\}$ denotes the set of nodes and E denotes the set of undirected edges so that $(i,j)\in E$ if nodes i and j share a physical connection. We assume that G forms a single connected component and denote by $\mathscr{N}(i)=\{j\in V\mid(i,j)\in E\}$ the set of neighbors of node i, i.e., those nodes j so that i and j can communicate directly. An example of such a network is given in Figure 7.1.

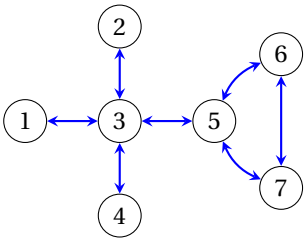


Figure 7.1: The communication graph G of a simple seven node network. Numbered circles denote nodes and their identifiers while the double ended arrows denote the undirected edges.

As previously mentioned, we are interested in using this network to perform distributed convex optimization. In this way, assume that each node is equipped with a function $f_i \in \Gamma_0\left(\mathbb{R}^M\right)\ \forall i \in V$ parameterized by a local variable $\mathbf{x}_i \in \mathbb{R}^M$, i.e. that the local variables at each node all have the same dimensionality, where, Γ_0 denotes the family of closed, convex and proper (CCP) functions. Based on these definitions, our objective is to solve the following consensus problem in a distributed manner,

$$\min_{\mathbf{x}_i\ \forall\, i \in V} \sum_{i \in V} f_i(\mathbf{x}_i) \quad \text{s.t}\ a_{i|j}\mathbf{x}_i + a_{j|i}\mathbf{x}_j = \mathbf{0} \quad \forall\ (i,j) \in E, \tag{7.1}$$

where the each scalar $a_{i|j}$ is defined as

$$a_{i|j} = \begin{cases} 1 & \text{if } i > j \\ -1 & \text{otherwise} \end{cases},$$

and the subscript $i|j$ is a directed edge identifier denoting the edge from node i to node j. The constraints of (7.1) ensure consensus between neighboring nodes within the network where we also assume that (7.1) is feasible, i.e., $\bigcap\limits_{i \in V} \text{dom}(f_i) \neq \emptyset$.

For this work, we consider the case where the underlying network topology is allowed to vary with time. In particular for each time step k, we will assume that there exists a simple, undirected, connected graph $G_{(k)}\left(V, E_{(k)}\right)$ that encapsulates the structure of the network. In this way, the set of nodes in the network does not vary with time while the edge sets $E^{(k)}$ can vary. The objective function of (7.1) is therefore the same for all network instances while the constraint functions simply enforce consensus between the local variables at each node. This assumption in turn ensures that the set of minimizers of (7.1), denoted by $\mathbf{X}^*$, does not change with the network topology and is constant across all time steps. This particular point raises the question of whether we can solve (7.1) in a distributed fashion even in the presence of time varying connectivity. As we will show in the coming sections, the answer is yes.

7.3.2. EXPLOITING SEPARABILITY VIA LAGRANGIAN DUALITY

Given the prototype problem in (7.1), the design of our distributed optimization solver echoes our other efforts within this thesis by aiming to address the coupling between the primal variables $\mathbf{x}_i$ at each node due to the linear constraint functions. For now we will consider the case of a fixed network topology but will return to the time varying network case in Section sec:ch7:timevarying. As with many classic approaches in the literature, we can exploit the separability of (7.1) via Lagrangian duality to overcome the aforementioned variable coupling. In particular, the Lagrangian of (7.1) is given by

$$\mathscr{L}\left(\{\mathbf{x}_i \mid i \in V\}, \{\boldsymbol{\nu}_{ij} \mid (i,j) \in E\}\right) = \sum_{i \in V} f_i(\mathbf{x}_i) - \sum_{(i,j) \in E} \boldsymbol{\nu}_{ij}^T \left(a_{i|j}\mathbf{x}_i + a_{j|i}\mathbf{x}_j\right), \tag{7.2}$$

where the set $\mathscr{N}(i) = \{j \in V \mid (i,j) \in E\}$ denotes the neighborhood of node i and $\boldsymbol{\nu}_{ij} \in \mathbb{R}^M$ is the vector of dual variables associated with the constraints along the undirected edge (i,j). Note the distinction between the undirected edge identifier ij and directed identifier $i|j$.

To exploit the separability of (7.2), we can therefore consider solving (7.1) in the dual domain. Specifically, the Lagrange dual problem of (7.1) is given by

$$\min_{\boldsymbol{\nu}} \quad \sum_{i\in V} f_i^*\left(\sum_{j\in\mathcal{N}(i)} a_{i|j}\boldsymbol{\nu}_{ij}\right), \tag{7.3}$$

where f_i^* is the Fenchel conjugate of f_i. By inspection, the resulting problem is still separable over the set of nodes but unfortunately each $\boldsymbol{\nu}_{ij}$ in (7.3) is utilized in two conjugate functions, f_i^* and f_j^*, resulting in a coupling between neighboring nodes.

To decouple the objective terms, we can lift the dimension of the dual problem by introducing copies of each $\boldsymbol{\nu}_{ij}$ at nodes i and j. The pairs of additional directed edge variables are denoted by $\boldsymbol{\lambda}_{i|j}, \boldsymbol{\lambda}_{j|i}$ $\forall (i,j) \in E$ and are associated with nodes i and j respectively. Specifically, we substitute $\boldsymbol{\nu}_{ij}$ with $a_{i|j}\boldsymbol{\lambda}_{i|j}$ at node i and $\boldsymbol{\nu}_{ij}$ with $a_{j|i}\boldsymbol{\lambda}_{j|i}$ at node j. To ensure equivalence of the problems, these are constrained so that at optimality $a_{i|j}\boldsymbol{\lambda}_{i|j} = a_{j|i}\boldsymbol{\lambda}_{j|i}$ or equivalently that $\boldsymbol{\lambda}_{i|j} + \boldsymbol{\lambda}_{j|i} = \mathbf{0}$ as $a_{i|j} = a_{j|i}$ $\forall (i,j) \in E$. The resulting problem is referred to as the *extended dual* of Eq (7.1) and is given by

$$\min_{\boldsymbol{\lambda}} \sum_{i\in V} f_i^*\left(\sum_{j\in\mathcal{N}(i)} \boldsymbol{\lambda}_{i|j}\right) \quad \text{s.t. } \boldsymbol{\lambda}_{i|j} + \boldsymbol{\lambda}_{j|i} = \mathbf{0} \quad \forall i \in V, j \in \mathcal{N}(i). \tag{7.4}$$

The proposed lifted problem is appealing from the perspective of alternating minimization techniques as it can be partitioned into two sections: a fully node separable objective function and a set of edge based constraints.

From (7.4) we can note that the the lifted dual is dependent on a set of dual variables which scale with the topology of the network, i.e., the extended dual problem has different dimensions for different instances of G. While this lifting naturally leads to a distributed implementation in time invariant networks, the dependence of dual variable dimensionality on network structure limits the use of traditional monotone operator based methods. This is despite the fact that the primal optimal variables $\mathbf{X}^*$ are independent of network topology. In the coming sections, we introduce a specific topology dependent change of variables to overcome this challenge, ultimately leading to a distributed consensus algorithm for use in time varying networks in Section 7.6.

7.3.3. SIMPLIFYING NOTATION

To simplify our notation in the coming analysis, in this section we introduce the following compact vector notation for Eq. (7.4). Specifically we show that (7.4) can be rewritten as

$$\min_{\boldsymbol{\lambda}} f^*(\mathbf{C}^T\boldsymbol{\lambda}) \quad \text{s.t. } (\mathbf{I}+\mathbf{P})\,\boldsymbol{\lambda} = \mathbf{0}. \tag{7.5}$$

DUAL VECTOR NOTATION

The basis of this simplified notation is the introduction of the dual vector $\boldsymbol{\lambda}$. Specifically, $\boldsymbol{\lambda}$ is a stacked vector of all $\boldsymbol{\lambda}_{i|j}$ where the ordering of this stacking is given by $1|2 < 1|3 < \cdots < 1|N < 2|1 < 2|3 < \cdots < N|N-1$. In the case of a fully connected network the vector $\boldsymbol{\lambda} \in \mathbb{R}^{M_E}$ is given by

$$\boldsymbol{\lambda} = \left[\boldsymbol{\lambda}_{1|2}^T, \cdots, \boldsymbol{\lambda}_{1|N}^T, \boldsymbol{\lambda}_{2|1}^T, \cdots, \boldsymbol{\lambda}_{N|N-1}^T\right]^T. \tag{7.6}$$

In the case of other network topologies which are not fully connected, non-existent edge variables can simply be omitted. In the following, the scalar $M_E = \sum_{i \in V} |\mathcal{N}(i)| M = 2|E|M$ denotes the total number of dual variables in (7.4).

COMPACT OBJECTIVE NOTATION

Given the definition of the dual vector $\boldsymbol{\lambda}$, we now move to simplifying the objective function. Firstly, we define the global function

$$f : \mathbb{R}^{M_V} \mapsto \mathbb{R}, \mathbf{x} \mapsto \sum_{i \in V} f_i(\mathbf{x}_i), \tag{7.7}$$

as the sum of all local functions where $\mathbb{R}^{M_V} = \mathbb{R}^M \times \mathbb{R}^M \times ... \times \mathbb{R}^M$, the scalar $M_V = NM$ denotes the total number of node variables. Similarly, f^*, which denotes the conjugate function of f has the same structure, i.e. $f^* : \mathbb{R}^{M_V} \mapsto \mathbb{R}, \mathbf{v} \mapsto \sum_{i \in V} f_i^*(\mathbf{v}_i)$.

We can then define a matrix $\mathbf{C} \in \mathbb{R}^{M_E \times M_V}$ to rewrite our objective using $\boldsymbol{\lambda}$ and f. In the case of a fully connected network these terms are given by

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_1 & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{C}_N \end{bmatrix} \otimes \mathbf{I}_M, \mathbf{C}_i = \mathbf{1}_{|\mathcal{N}(i)|} \quad \forall i \in V, \tag{7.8}$$

where $\mathbf{1}_{|\mathcal{N}(i)|}$ and $\mathbf{I}_M$ are a $|\mathcal{N}(i)| \times 1$ ones vector and $M \times M$ identity matrix respectively, and the symbol $\otimes$ denotes the Kronecker product of two matrices. Combining (7.6), (7.7) and (7.8), the objective function of Eq. (7.4) can be equivalently written as

$$f^*(\mathbf{C}^T \boldsymbol{\lambda}),$$

where the term $\mathbf{C}^T \boldsymbol{\lambda}$ produces a stacked set of M-dimensional vectors, one at each node i, which are the sum of local directed edge variables.

COMPACT CONSTRAINTS NOTATION

Similarly to the objective, we can define an additional matrix to rewrite the constraint functions using our vector notation. For this task we introduce the symmetric permutation matrix $\mathbf{P} \in \mathbb{R}^{M_E \times M_E}$ that permutes each pair of variables $\boldsymbol{\lambda}_{i|j}$ and $\boldsymbol{\lambda}_{j|i}$. This allows the constraints in (7.4) to be rewritten as $(\mathbf{I} + \mathbf{P})\boldsymbol{\lambda} = \mathbf{0}$. The vector $\boldsymbol{\lambda}$ is therefore only feasible if it is contained in $\ker(\mathbf{I} + \mathbf{P})$. Using this compact notation, we are now ready to continue with our derivation.

7.3.4. MODIFYING THE EXTENDED DUAL VIA A CHANGE OF VARIABLES

At this stage we could directly move to deriving a distributed solver for the case of a time invariant network. However, to accommodate time varying network topologies in the later portion of this chapter, we instead make a small modification to (7.5) in the form of a change of variables given by $\boldsymbol{\lambda} = \boldsymbol{\Phi}\tilde{\boldsymbol{\lambda}}$ where the matrix $\boldsymbol{\Phi} \succ \mathbf{0}$. In this way, (7.5) can be rewritten as

$$\min_{\tilde{\boldsymbol{\lambda}}} f^*(\mathbf{C}^T \boldsymbol{\Phi} \tilde{\boldsymbol{\lambda}}) \quad \text{s.t } (\mathbf{I} + \mathbf{P})\boldsymbol{\Phi}\tilde{\boldsymbol{\lambda}} = \mathbf{0}. \tag{7.9}$$

In particular, we define $\mathbf{\Phi}$ as being a diagonal matrix such that

$$\mathbf{\Phi} = \begin{bmatrix} \mathbf{\Phi}_1 & \dots & \mathbf{0} \\ \vdots & \ddots & \mathbf{\Phi}_N \end{bmatrix} \otimes \mathbf{I}_M, \tag{7.10}$$

where the submatrices $\mathbf{\Phi}_i$ are given by

$$\mathbf{\Phi}_i = \frac{1}{\sqrt{|\mathcal{N}(i)|}} \mathbf{I}_{|\mathcal{N}(i)|} \; \forall i \in V,$$

where we can note that each $\boldsymbol{\phi}_i \succ \mathbf{0}$ due to the connected assumption of the underlying network. Furthermore, this assumption will hold for all such networks.

The motivation behind this change of variables is as follows. From (7.4), it is clear that the topology of the network influences the structure of the extended dual, notably that the total number of extended dual variables changes for each network. Specifically, at each node i, the associated set of variables $\lambda_{i|j}$ are implicitly dependent on $|\mathcal{N}(i)|$. Defining $\mathbf{\Phi}$ as per (7.10) aims to remove this dependency. For instance, considering the definition of $\mathbf{C}$, it follows that the matrix product $\mathbf{C}^T\mathbf{\Phi}\mathbf{\Phi}\mathbf{C}$ satisfies

$$\begin{aligned} \mathbf{C}^T\mathbf{\Phi}\mathbf{\Phi}\mathbf{C} &= \left(\begin{bmatrix} \mathbf{C}_1 & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{C}_N \end{bmatrix}^T \begin{bmatrix} \mathbf{\Phi}_1\mathbf{\Phi}_1 & \dots & \mathbf{0} \\ \vdots & \ddots & \mathbf{\Phi}_N\mathbf{\Phi}_N \end{bmatrix} \begin{bmatrix} \mathbf{C}_1 & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{C}_N \end{bmatrix} \right) \otimes \mathbf{I}_M \\ &= \begin{bmatrix} \frac{\mathbf{1}^T_{|\mathcal{N}(1)|}\mathbf{1}_{|\mathcal{N}(1)|}}{|\mathcal{N}(1)|} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \frac{\mathbf{1}^T_{|\mathcal{N}(N)|}\mathbf{1}_{|\mathcal{N}(N)|}}{|\mathcal{N}(N)|} \end{bmatrix} \otimes \mathbf{I}_M = \mathbf{I}_N \otimes \mathbf{I}_M = \mathbf{I}_{M_V}, \end{aligned} \tag{7.11}$$

where the first equality uses the mixed-product property of Kronecker products. In Section 7.4, we show that this equality is a key step in removing the dependency of our proposed distributed algorithm on network topology. In this way, our particular choice of $\mathbf{\Phi}$ ultimately allows us to form an algorithm where, like the primal variables, the set of optimal dual variables also do not vary for different network topologies.

7.3.5. MONOTONIC INCLUSIONS AND FIXED POINT PROBLEMS

To solve (7.9) in a distributed fashion, we can firstly rewrite it in an equivalent unconstrained form given by

$$\min_{\tilde{\boldsymbol{\lambda}}} \quad f^*(\mathbf{C}^T\mathbf{\Phi}\tilde{\boldsymbol{\lambda}}) + \iota_{\ker((\mathbf{I}+\mathbf{P})\mathbf{\Phi})}(\tilde{\boldsymbol{\lambda}}), \tag{7.12}$$

where $\iota_{\ker(\mathbf{I}+\mathbf{P})}$ is the indicator function of the edge based constraints defined as

$$\iota_{\ker((\mathbf{I}+\mathbf{P})\mathbf{\Phi})}(\mathbf{y}) = \begin{cases} 0 & (\mathbf{I}+\mathbf{P})\mathbf{\Phi}\mathbf{y} = \mathbf{0} \\ +\infty & \text{otherwise.} \end{cases}$$

Under our assumptions that f is closed, convex and proper and the feasibility of (7.1), a minimizer of (7.12) can be equivalently found as a zero point of its subdifferential

which is given by

$$\mathbf{0} \in \mathbf{\Phi C} \partial f^* \left(\mathbf{C}^T \mathbf{\Phi}^T \tilde{\boldsymbol{\lambda}} \right) + \partial \iota_{\ker((\mathbf{I}+\mathbf{P})\mathbf{\Phi})} \left(\tilde{\boldsymbol{\lambda}} \right). \tag{7.13}$$

The operators $\mathbf{T}_1 = \mathbf{\Phi C} \partial f^* \left(\mathbf{C}^T \mathbf{\Phi}^T \right)$ and $\mathbf{T}_2 = \partial \iota_{\ker((\mathbf{I}+\mathbf{P})\mathbf{\Phi})}$ are by design separable over the set of nodes and edges respectively. The operators ∂f^* and $\partial \iota_{\ker((\mathbf{I}+\mathbf{P})\mathbf{\Phi})}$ are the subdifferentials of CCP functions and thus are maximal monotone. It follows that a zero-point of (7.13) can be found via a variety of operator splitting methods.

In the following we consider the use of Peaceman-Rachford splitting to solve (7.13). However, other approaches such as Forward-Backward splitting, Primal-Dual splitting methods and more could be used for this purpose as well. Peaceman-Rachford splitting allows for the rephrasing of the monotonic inclusion (7.13) as a fixed point problem given by

$$\mathbf{R}_{\mathbf{T}_2,\rho} \circ \mathbf{R}_{\mathbf{T}_1,\rho} (\mathbf{z}) = \mathbf{z}, \quad \tilde{\boldsymbol{\lambda}} = \mathbf{J}_{\mathbf{T}_1,\rho} (\mathbf{z}) \tag{7.14}$$

where $\mathbf{R}_{\mathbf{T}_i,\rho}$ and $\mathbf{J}_{\mathbf{T}_i,\rho}$ are the reflected resolvent and resolvent operators of $\mathbf{T}_i$ respectively and $\rho > 0$ is a non-negative step size. As both $\mathbf{T}_1$ and $\mathbf{T}_2$ are maximal monotone operators, it follows that both $\mathbf{R}_{\mathbf{T}_1,\rho}$ and $\mathbf{R}_{\mathbf{T}_2,\rho}$ are nonexpansive and thus so too is their composition $\mathbf{R}_{\mathbf{T}_2,\rho} \circ \mathbf{R}_{\mathbf{T}_1,\rho}$. The introduced $\mathbf{z} \in \mathbb{R}^{M_E}$ variable will be referred to as an *auxiliary* variable. Like the extended dual variables in Section 7.3.2, here the dimensionality of the auxiliary variables also varies with the topology of the network. While sufficient for the time invariant case, it is this variability which poses challenges in the formation of monotone operator based solvers for use in time varying networks. Fortunately this will be addressed in the coming sections.

The fixed point problem in (7.14) can be solved via classic iterative methods for nonexpansive operators. We consider the use of an α-averaged Banach-Picard iteration (although any Krasnosel'skiĭ-Mann iteration could be used as well). The resulting iterative scheme is given by

$$\mathbf{z}^{(k+1)} = (1-\alpha)\mathbf{z}^{(k)} + \alpha \mathbf{R}_{\mathbf{T}_2,\rho} \circ \mathbf{R}_{\mathbf{T}_1,\rho} \left(\mathbf{z}^{(k)} \right),$$

where $\alpha \in (0,1)$.

7.3.6. DISTRIBUTED ALGORITHM IMPLEMENTATION

Before we can understand how the additional matrix $\mathbf{\Phi}$ assists in the formation of a distributed solver for time-varying networks we first need to address the evaluation of the reflected resolvents $\mathbf{R}_{\mathbf{T}_1,\rho}$ and $\mathbf{R}_{\mathbf{T}_2,\rho}$. These two procedures are summarized in the following two Lemmas.

Lemma 7.3.1. $\mathbf{y}^{(k+1)} = \mathbf{R}_{\mathbf{T}_1,\rho} \left(\mathbf{z}^{(k)} \right)$ *can be computed as*

$$\begin{aligned}
\mathbf{x}^{(k+1)} &= \arg\min_{\mathbf{x}} \left(f(\mathbf{x}) - \left\langle \mathbf{C}^T \mathbf{\Phi}^T \mathbf{z}^{(k)}, \mathbf{x} \right\rangle + \frac{\rho}{2} ||\mathbf{\Phi C x}||^2 \right) \\
\tilde{\boldsymbol{\lambda}}^{(k+1)} &= \mathbf{z}^{(k)} - \rho \mathbf{\Phi C} \mathbf{x}^{(k+1)} \\
\mathbf{y}^{(k+1)} &= 2\tilde{\boldsymbol{\lambda}}^{(k+1)} - \mathbf{z}^{(k)} = \mathbf{z}^{(k)} - 2\rho \mathbf{\Phi C} \mathbf{x}^{(k+1)}
\end{aligned}$$

A proof of this result can be found in Appendix 7.A.

Lemma 7.3.2. $\mathbf{w}^{(k+1)} = \mathbf{R}_{\mathbf{T}_2,\rho}\left(\mathbf{y}^{(k+1)}\right)$ *can be computed as*

$$\mathbf{w}^{(k+1)} = \mathscr{K}\mathbf{y}^{(k+1)} - \mathscr{R}\mathbf{y}^{(k+1)},$$

where $\mathscr{K} = \prod_{\ker((\mathbf{I}+\mathbf{P})\mathbf{\Phi})}$, $\mathscr{R} = \mathbf{I} - \mathscr{K}$ *and the operator* $\prod_{\mathscr{A}}$ *denotes the orthogonal projection onto the subspace* $\mathscr{A}$.

The proof for this result is included in Appendix 7.B. Utilizing Lemmas 7.3.1 and 7.3.2 and recalling from (7.11) that for our particular choice of $\mathbf{\Phi}$ that $\mathbf{C}^T\mathbf{\Phi}\mathbf{\Phi}^T\mathbf{C} = \mathbf{I}_{M_V}$, it therefore follows that (7.14) can be computed via Algorithm 12. The convergence of this algorithm follows from the properties of averaged operators which, due to the finite dimensionality of the problems considered, guarantees convergence of the auxiliary variables to a fixed point (see [34, Theorem 5.15]).

Algorithm 12 Distributed Consensus

Initialise: $\mathbf{z}^{(0)} \in \mathbb{R}^{M_E}$
for $k = 0, \cdots,$ **do**
  $\mathbf{x}^{(k+1)} = \operatorname{argmin}_{\mathbf{x}}\left(f(\mathbf{x}) - \left\langle \mathbf{C}^T\mathbf{\Phi}\mathbf{z}^{(k)}, \mathbf{x}\right\rangle + \frac{\rho}{2}||\mathbf{x}||^2\right)$
  $\mathbf{y}^{(k+1)} = \mathbf{z}^{(k)} - 2\rho\mathbf{\Phi}\mathbf{C}\mathbf{x}^{(k+1)}$
  $\mathbf{w}^{(k+1)} = \mathscr{K}\mathbf{y}^{(k+1)} - \mathscr{R}\mathbf{y}^{(k+1)}$
  $\mathbf{z}^{(k+1)} = (1-\alpha)\mathbf{z}^{(k)} + \alpha\mathbf{w}^{(k+1)}$
end for

7.4. DISTRIBUTED CONSENSUS IN TIME INVARIANT NETWORKS

From the perspective of time varying consensus, the main problem with Algorithm 12 is the dependence of the dimensionality of many of the variables $\left(\mathbf{y}^{(k+1)}, \mathbf{w}^{(k+1)}, \mathbf{z}^{(k+1)}\right)$ on the number of edges in the network which may change at each iteration. In contrast, the dimensionality of the primal variables only depends on the number of nodes in the network which, in our time varying model, is constant. Moreover, we are ultimately interested in using this algorithm to compute the optimal $\mathbf{x}$ variables which are of much lower dimensionality than either the dual or the auxiliary variables. This suggests that there is redundancy in the way the algorithm represents its variables. Furthermore, the fact that the dual and auxiliary are generated via linear operations suggests that this redundancy can be eliminated. In this section we aim to address this point by demonstrating an equivalent set of update equations whose dimensionality do not depend on the connectivity of the network but only on the number of nodes. In particular, we demonstrate this equivalent form for the simpler time invariant case which we then extend to the time varying case in Section 7.6.

7.4.1. REMOVING THE DEPENDENCE ON THE AUXILIARY VARIABLES

To remove our dependence on network connectivity, we firstly note that by splitting the auxiliary $\mathbf{z}$ variables into two components $\mathbf{z}_K^{(k)} = \mathscr{K}\mathbf{z}^{(k)}$ and $\mathbf{z}_R^{(k)} = \mathscr{R}\mathbf{z}^{(k)}$, Algorithm 12

Algorithm 13 Split Distributed Consensus

Initialise: $\mathbf{z}^{(0)} \in \mathbb{R}^{M_E}$
for $k = 0, \cdots,$ **do**

$$\mathbf{x}^{(k+1)} = \underset{\mathbf{x}}{\operatorname{argmin}} \left(f(\mathbf{x}) - \left\langle \begin{bmatrix} \mathscr{K}\mathbf{\Phi C} \\ \mathscr{R}\mathbf{\Phi C} \end{bmatrix}^T \begin{bmatrix} \mathbf{z}_K^{(k)} \\ \mathbf{z}_R^{(k)} \end{bmatrix}, \mathbf{x} \right\rangle + \tfrac{\rho}{2} ||\mathbf{x}||^2 \right)$$

$$\mathbf{y}^{(k+1)} = \begin{bmatrix} \mathbf{z}_K^{(k)} \\ \mathbf{z}_R^{(k)} \end{bmatrix} - 2\rho \begin{bmatrix} \mathscr{K}\mathbf{\Phi C} \\ \mathscr{R}\mathbf{\Phi C} \end{bmatrix} \mathbf{x}^{(k+1)}$$

$$\mathbf{w}^{(k+1)} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I} \end{bmatrix} \mathbf{y}^{(k+1)}$$

$$\begin{bmatrix} \mathbf{z}_K^{(k+1)} \\ \mathbf{z}_R^{(k+1)} \end{bmatrix} = (1-\alpha) \begin{bmatrix} \mathbf{z}_K^{(k)} \\ \mathbf{z}_R^{(k)} \end{bmatrix} + \alpha \mathbf{w}^{(k+1)}$$

end for

can be equivalently expressed as Algorithm 13. As in Lemma 7.3.2, here $\mathscr{K} = \underset{\ker((\mathbf{I}+\mathbf{P})\mathbf{\Phi})}{\Pi}$, while $\mathscr{R} = \mathbf{I} - \mathscr{K}$.

Like Algorithm 12, due to nonexpansiveness and the use of operator averaging, this split form is averaged in the stacked variables $\left[\mathbf{z}_K^{(k+1)T}, \mathbf{z}_R^{(k+1)T}\right]^T$ and thus converges to a fixed point asymptotically .

Using the split structure in Algorithm 13, we can demonstrate a compact form of distributed consensus which is independent of the number of edges of the network. To achieve this point, we firstly define the additional variables

$$\boldsymbol{\gamma}_K^{(k+1)} = \mathbf{C}^T\mathbf{\Phi}\mathbf{z}_K^{(k)}, \; \boldsymbol{\gamma}_R^{(k+1)} = \mathbf{C}^T\mathbf{\Phi}\mathbf{z}_R^{(k)}, \tag{7.15}$$

where the vectors $\boldsymbol{\gamma}_K^{(k)}, \boldsymbol{\gamma}_R^{(k)} \in \mathbb{R}^{M_V} \; \forall k \in \mathbb{N}$. We can then rewrite the primal $\mathbf{x}$ update from Algorithm 13 in the more compact form

$$\mathbf{x}^{(k+1)} = \underset{\mathbf{x}}{\operatorname{argmin}} \left(f(\mathbf{x}) - \left\langle \begin{bmatrix} \mathbf{I} \\ \mathbf{I} \end{bmatrix}^T \begin{bmatrix} \boldsymbol{\gamma}_K^{(k)} \\ \boldsymbol{\gamma}_R^{(k)} \end{bmatrix}, \mathbf{x} \right\rangle + \frac{\rho}{2} ||\mathbf{x}||^2 \right). \tag{7.16}$$

Other than the computation of the primal $\mathbf{x}$ update equations of Algorithm 13, all remaining operations are linear and thus can be combined to form a single update equation given by

$$\begin{bmatrix} \mathbf{z}_K^{(k+1)} \\ \mathbf{z}_R^{(k+1)} \end{bmatrix} = \begin{bmatrix} \mathbf{z}_K^{(k)} \\ (1-2\alpha)\mathbf{z}_R^{(k)} \end{bmatrix} + 2\rho\alpha \begin{bmatrix} -\mathscr{K}\mathbf{\Phi C} \\ \mathscr{R}\mathbf{\Phi C} \end{bmatrix} \mathbf{x}^{(k+1)}.$$

Combining with the definition of the $\boldsymbol{\gamma}$ variables in (7.15), it follows that

$$\begin{bmatrix} \boldsymbol{\gamma}_K^{(k+1)} \\ \boldsymbol{\gamma}_R^{(k+1)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\gamma}_K^{(k)} \\ (1-2\alpha)\boldsymbol{\gamma}_R^{(k)} \end{bmatrix} + 2\rho\alpha \begin{bmatrix} -\mathbf{C}^T\mathbf{\Phi}\mathscr{K}\mathbf{\Phi C} \\ \mathbf{C}^T\mathbf{\Phi}\mathscr{R}\mathbf{\Phi C} \end{bmatrix} \mathbf{x}^{(k+1)}.$$

By defining $\tilde{\mathscr{L}} = \mathbf{C}^T\mathbf{\Phi}\mathscr{K}\mathbf{\Phi C}$ and recalling that $\mathscr{R} = \mathbf{I} - \mathscr{K}$, $\mathbf{C}^T\mathbf{C} = \mathbf{I}$, it follows that

$$\begin{bmatrix} \boldsymbol{\gamma}_K^{(k+1)} \\ \boldsymbol{\gamma}_R^{(k+1)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\gamma}_K^{(k)} \\ (1-2\alpha)\boldsymbol{\gamma}_R^{(k)} \end{bmatrix} + 2\rho\alpha \begin{bmatrix} -\tilde{\mathscr{L}} \\ \mathbf{I} - \tilde{\mathscr{L}} \end{bmatrix} \mathbf{x}^{(k+1)}. \tag{7.17}$$

As we will show in the following section, the matrix $\tilde{\mathscr{L}}$ corresponds to a particular weighted graph Laplacian and therefore defines a form of weighted averaging of data between connected nodes.

Combining (7.16) and (7.17), we can replace the update procedure in Algorithm 13 to form Algorithm 14. In doing so, we have removed the dependence of the dimensionality of all variables on the connectivity of the underlying network, as desired.

Algorithm 14 Time Invariant Distributed Consensus

1: **Initialise:** $\boldsymbol{\gamma}_R^{(0)}, \boldsymbol{\gamma}_N^{(0)} = \mathbf{0}$
2: **for** $k = 0, \cdots,$ **do**
3: $\quad \mathbf{x}^{(k+1)} = \underset{\mathbf{x}}{\operatorname{argmin}} \left(f(\mathbf{x}) - \left\langle \begin{bmatrix} \mathbf{I} \\ \mathbf{I} \end{bmatrix}^T \begin{bmatrix} \boldsymbol{\gamma}_K^{(k)} \\ \boldsymbol{\gamma}_R^{(k)} \end{bmatrix}, \mathbf{x} \right\rangle + \frac{\rho}{2} ||\mathbf{x}||^2 \right)$
4: $\quad \begin{bmatrix} \boldsymbol{\gamma}_K^{(k+1)} \\ \boldsymbol{\gamma}_R^{(k+1)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\gamma}_K^{(k)} \\ (1-2\alpha)\boldsymbol{\gamma}_R^{(k)} \end{bmatrix} + 2\rho\alpha \begin{bmatrix} -\tilde{\mathscr{L}} \\ \mathbf{I} - \tilde{\mathscr{L}} \end{bmatrix} \mathbf{x}^{(k+1)}$
5: **end for**

This particular representation is highly appealing as it casts distributed optimization as a composition of local optimizations at each node with a data fusion step based on the local connectivity of each node based on the matrix $\tilde{\mathscr{L}}$. This final operation corresponds to a weighted averaging of data between connected nodes in the network, highlighting the distributed nature of the algorithm. Notably, each node needs to only broadcast its current primal estimate to its neighbors at each time step, removing the need for the transmission of edge specific variables present in 13. The introduced $\boldsymbol{\gamma}$ variables also have far lower dimensionality than their edge based dual and auxiliary counterparts. Through the transformed update equations, we can draw a similarity with distributed subgradient methods [11], where a combination of local forward update steps is followed by a weighted averaging of variables.

In the following section we demonstrate that the particular weighted averaging performed by the matrix $\tilde{\mathscr{L}}$ is in fact a weighted graph Laplacian matrix which we use in Section 7.5 to prove the convergence of the time invariant method in Algorithm 14 to a primal optimal solution. This time invariant approach is extended in Section 7.6 to the case of time varying networks.

7.4.2. A WEIGHTED GRAPH LAPLACIAN MIXING MATRIX

We now move to demonstrating how the matrix $\tilde{\mathscr{L}} = \mathbf{C}^T \boldsymbol{\Phi} \mathscr{K} \boldsymbol{\Phi} \mathbf{C}$ corresponds to a weighted graph Laplacian. To this end we consider the following Lemma.

Lemma 7.4.1. *The matrix $\tilde{\mathscr{L}}$ represents a weighted graph Laplacian given by*

$$\tilde{\mathscr{L}} = \tilde{\mathbf{D}} - \tilde{\mathbf{W}} = \left(\hat{\mathbf{D}} - \hat{\mathbf{W}}\right) \otimes \mathbf{I}_M = \hat{\mathscr{L}} \otimes \mathbf{I}_M,$$

where

$$\begin{aligned} \tilde{\mathbf{W}} &= \left(\mathbf{W} \oslash \left(\mathbf{W}\mathbf{1}_N \mathbf{1}_N^T + \mathbf{1}_N \mathbf{1}_N^T \mathbf{W}\right)\right) \otimes \mathbf{I}_M = \hat{\mathbf{W}} \otimes \mathbf{I}_M \\ \tilde{\mathbf{D}} &= \operatorname{diag}\left(\tilde{\mathbf{W}}\mathbf{1}_{NM}\right) = \hat{\mathbf{D}} \otimes \mathbf{I}_M, \end{aligned}$$

where $\oslash$ and $\otimes$ denote the element-wise division and the Kronecker product of two matrices respectively. The matrix $\mathbf{W}$ denotes the adjacency matrix of $G(V,E)$.

The proof of this Lemma can be found in Appendix 7.C. Two useful properties of $\tilde{\mathscr{L}}$, which we will make use of in the future, are given in the following Lemmas.

Lemma 7.4.2. *The kernel of $\tilde{\mathscr{L}}$ is spanned by the columns of $\frac{1}{\sqrt{N}}\mathbf{1}\otimes\mathbf{I}_M$. In other words the columns of $\frac{1}{\sqrt{N}}\mathbf{1}\otimes\mathbf{I}_M$ are eigenvectors of $\tilde{\mathscr{L}}$ with corresponding eigenvalues 0.*

Lemma 7.4.3. *The eigenvalues of $\tilde{\mathscr{L}}$ are contained in $[0,1]$. Additionally, 1 is only an eigenvalue of $\tilde{\mathscr{L}}$ if G is a bipartite graph.*

The proof for these Lemmas can be found in Appendix 7.D and 7.E respectively. In Section 7.5, we use these properties to prove the convergence of the proposed method in time invariant networks.

7.4.3. OPTIMAL $\boldsymbol{\gamma}$ VARIABLES AND NETWORK TOPOLOGY

The reduced form of Algorithm 14 allows us to also demonstrate that the optimal introduced $\boldsymbol{\gamma}$ variables are invariant to changes in the network topology. Specifically, by considering the primal $\mathbf{x}$ updates, from the convexity of the minimized problem in (7.16), it follows that for each $\mathbf{x}$ update,

$$\mathbf{x}^{(k+1)} = \left(\rho\mathbf{I}+\partial f\right)^{-1}\begin{bmatrix}\mathbf{I}\\ \mathbf{I}\end{bmatrix}^T\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix},$$

or equivalently that

$$\mathbf{x}^{(k+1)} = \left(\mathbf{I}+\frac{\partial f}{\rho}\right)^{-1}\left(\frac{1}{\rho}\begin{bmatrix}\mathbf{I}\\ \mathbf{I}\end{bmatrix}^T\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}\right). \tag{7.18}$$

The operator $\left(\mathbf{I}+\frac{\partial f}{\rho}\right)^{-1}$ is a resolvent operator due to the maximal monotonicity of ∂f and is therefore single valued by nature. Defining the set of optimal $\boldsymbol{\gamma}$ variables by $\boldsymbol{\Gamma}^*$, it follows that

$$\forall\begin{bmatrix}\boldsymbol{\gamma}_K^*\\ \boldsymbol{\gamma}_R^*\end{bmatrix}\in\boldsymbol{\Gamma}^*\quad\exists\mathbf{x}^*\in\mathbf{X}^*\,\Big|\,\mathbf{x}^*=\left(\rho\mathbf{I}+\partial f\right)^{-1}\begin{bmatrix}\mathbf{I}\\ \mathbf{I}\end{bmatrix}^T\begin{bmatrix}\boldsymbol{\gamma}_K^*\\ \boldsymbol{\gamma}_R^*\end{bmatrix}.$$

It therefore follows that, like $\mathbf{X}^*$, this $\boldsymbol{\Gamma}^*$ does not vary with the underlying network topology. As alluded to in Section 7.3.4, this consistency stems from our particular choice of the topology dependent metric $\boldsymbol{\Phi}$.

7.5. CONVERGENCE IN TIME INVARIANT NETWORKS

While in the time-invariant network case the convergence of the method presented in Algorithm 14 follows from its relationship to averaged Peaceman-Rachford splitting, we can also show convergence directly by considering the introduced $\boldsymbol{\gamma}$ variables. The following analysis is also a useful stepping stone in understanding the convergence characteristics in the more complicated time-varying context in Section 7.6.

To demonstrate convergence, we firstly consider an equivalent form of the $\boldsymbol{\gamma}$ updates of Algorithm 14 given by

$$\begin{bmatrix}\boldsymbol{\gamma}_K^{(k+1)}\\ \boldsymbol{\gamma}_R^{(k+1)}\end{bmatrix} = (1-\alpha)\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix} + \alpha\begin{bmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{0} & -\mathbf{I}\end{bmatrix}\left(\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix} - 2\rho\begin{bmatrix}-\tilde{\mathscr{L}}\\ \mathbf{I}-\tilde{\mathscr{L}}\end{bmatrix}\mathbf{x}^{(k+1)}\right). \tag{7.19}$$

Additionally, we require the use of the inner product $\langle \mathbf{x},\mathbf{y}\rangle_{\boldsymbol{\theta}} = \langle \mathbf{x},\boldsymbol{\theta}\mathbf{y}\rangle$ and induced norm $\|\mathbf{x}\|_{\boldsymbol{\theta}}^2 = \langle \mathbf{x},\boldsymbol{\theta}\mathbf{x}\rangle$, both of which are parameterized by the matrix

$$\boldsymbol{\theta} = \begin{bmatrix}\tilde{\mathscr{L}}^{\dagger} & \mathbf{0}\\ \mathbf{0} & \left(\mathbf{I}-\tilde{\mathscr{L}}\right)^{\dagger}\end{bmatrix},$$

where $\bullet^{\dagger}$ denotes the Moore-Penrose pseudo inverse. Note that despite the fact that $\tilde{\mathscr{L}}$ and $\mathbf{I}-\tilde{\mathscr{L}}$ may be low rank, $\|\bullet\|_{\boldsymbol{\theta}}$ still defines a valid metric for all iterations as the stacked $\boldsymbol{\gamma}$ variables are contained within the range space of $\boldsymbol{\theta}$ by definition. Specifically, $\boldsymbol{\gamma}_K \in \operatorname{ran}\left(\tilde{\mathscr{L}}\right)$ and $\boldsymbol{\gamma}_R \in \operatorname{ran}\left(\mathbf{I}-\tilde{\mathscr{L}}\right)$ for all iterations. Using this metric, we can show that the $\boldsymbol{\gamma}$ variables are averaged and thus converge. To see this, consider the squared distance measure

$$\begin{aligned}\left\|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k+1)}\\ \boldsymbol{\gamma}_R^{(k+1)}\end{bmatrix} - \begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}\right\|_{\boldsymbol{\theta}}^2 &= \left\|(1-\alpha)\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix} + \alpha\begin{bmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{0} & -\mathbf{I}\end{bmatrix}\left(\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix} - 2\rho\begin{bmatrix}\tilde{\mathscr{L}}\\ \mathbf{I}-\tilde{\mathscr{L}}\end{bmatrix}\mathbf{x}^{(k+1)}\right) - \begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}\right\|_{\boldsymbol{\theta}}^2\\
&= (1-\alpha)\left\|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix} - \begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}\right\|_{\boldsymbol{\theta}}^2 + \alpha\left\|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix} - \begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix} - 2\rho\begin{bmatrix}\tilde{\mathscr{L}}\\ \mathbf{I}-\tilde{\mathscr{L}}\end{bmatrix}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^{*}\right)\right\|_{\boldsymbol{\theta}}^2\\
&\quad -4\alpha(1-\alpha)\left\|\begin{bmatrix}\mathbf{0}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix} - \begin{bmatrix}\mathbf{0}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix} - \rho\begin{bmatrix}\tilde{\mathscr{L}}\\ \mathbf{I}-\tilde{\mathscr{L}}\end{bmatrix}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^{*}\right)\right\|_{\boldsymbol{\theta}}^2,\end{aligned} \tag{7.20}$$

where in the first line we have used the equivalent $\boldsymbol{\gamma}$ update in (7.19) while in the second line we have used [34, Corollary 2.15] and the fact that $\begin{bmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{0} & -\mathbf{I}\end{bmatrix}$ is a nonexpansive operator for any quadratic form. It follows that if

$$\left\|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix} - \begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix} - 2\rho\begin{bmatrix}\tilde{\mathscr{L}}\\ \mathbf{I}-\tilde{\mathscr{L}}\end{bmatrix}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^{*}\right)\right\|_{\boldsymbol{\theta}}^2 \leq \left\|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix} - \begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}\right\|_{\boldsymbol{\theta}}^2, \tag{7.21}$$

then the sequence $\left(\left\|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix} - \begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}\right\|_{\boldsymbol{\theta}}^2\right)_{k\in\mathbb{N}}$ is non-increasing. To show that (7.21) holds, we begin by noting that

$$\begin{aligned}&\left\|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix} - \begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix} - 2\rho\begin{bmatrix}\tilde{\mathscr{L}}\\ \mathbf{I}-\tilde{\mathscr{L}}\end{bmatrix}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^{*}\right)\right\|_{\boldsymbol{\theta}}^2 = \left\|2\rho\begin{bmatrix}\tilde{\mathscr{L}}\\ \mathbf{I}-\tilde{\mathscr{L}}\end{bmatrix}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^{*}\right)\right\|_{\boldsymbol{\theta}}^2\\
&-2\left\langle 2\rho\begin{bmatrix}\tilde{\mathscr{L}}\\ \mathbf{I}-\tilde{\mathscr{L}}\end{bmatrix}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^{*}\right), \begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix} - \begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}\right\rangle_{\boldsymbol{\theta}} + \left\|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix} - \begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}\right\|_{\boldsymbol{\theta}}^2\\
&= 4\left\|\rho\left(\mathbf{x}^{(k+1)}-\mathbf{x}^{*}\right)\right\|^2 - 4\left\langle \rho\left(\mathbf{x}^{(k+1)}-\mathbf{x}^{*}\right), \begin{bmatrix}\mathbf{I}\\ \mathbf{I}\end{bmatrix}^{T}\left(\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix} - \begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}\right)\right\rangle + \left\|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix} - \begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}\right\|_{\boldsymbol{\theta}}^2,\end{aligned} \tag{7.22}$$

where in the third line we had used the definition of the inner product and induced norm $\langle \mathbf{x},\mathbf{y}\rangle_{\boldsymbol{\theta}}$ and $\|\mathbf{x}\|_{\boldsymbol{\theta}}^2$ and the fact that $\boldsymbol{\gamma}_K^{(k)} \in \operatorname{ran}(\tilde{\mathscr{L}})$, $\boldsymbol{\gamma}_R^{(k)} \in \operatorname{ran}(\mathbf{I}-\tilde{\mathscr{L}})$ $\forall k \in \mathbb{N}$. From (7.18), recall that

$$\mathbf{x}^{(k+1)} = \left(\mathbf{I}+\frac{\partial f}{\rho}\right)^{-1}\left(\frac{1}{\rho}\begin{bmatrix}\mathbf{I}\\ \mathbf{I}\end{bmatrix}^T\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}\right).$$

where the operator $\left(\mathbf{I}+\frac{\partial f}{\rho}\right)^{-1}$ is a resolvent and thus firmly nonexpansive which is defined as follows.

Definition 7.5.1. *Firmly Nonexpansive Operators: An operator* $\mathbf{T}:\mathscr{X}\to\mathscr{Y}$ *is firmly nonexpansive if* $\forall \mathbf{x}_1,\mathbf{x}_2 \in \mathscr{X}$, $\mathbf{y}_1 \in \mathbf{T}(\mathbf{x}_1)$, $\mathbf{y}_2 \in \mathbf{T}(\mathbf{x}_2)$

$$\|\mathbf{y}_1-\mathbf{y}_2\|^2 \le \langle \mathbf{y}_1-\mathbf{y}_2,\mathbf{x}_1-\mathbf{x}_2\rangle.$$

Using the structure of (7.18), it follows that

$$\|\mathbf{x}^{(k+1)}-\mathbf{x}^*\|^2 \le \left\langle \mathbf{x}^{(k+1)}-\mathbf{x}^*, \frac{1}{\rho}\begin{bmatrix}\mathbf{I}\\ \mathbf{I}\end{bmatrix}^T\left(\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}-\begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}\right)\right\rangle, \tag{7.23}$$

which, when combined with (7.22) and (7.23), ensures that

$$\left\|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}-\begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}-2\rho\begin{bmatrix}\tilde{\mathscr{L}}\\ \mathbf{I}-\tilde{\mathscr{L}}\end{bmatrix}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^*\right)\right\|_{\boldsymbol{\theta}}^2 \le \left\|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}-\begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}\right\|_{\boldsymbol{\theta}}^2. \tag{7.24}$$

By substituting (7.24) into the final equality of (7.20), we find that

$$\left\|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k+1)}\\ \boldsymbol{\gamma}_R^{(k+1)}\end{bmatrix}-\begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}\right\|_{\boldsymbol{\theta}}^2 \le \left\|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}-\begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}\right\|_{\boldsymbol{\theta}}^2 -4\alpha(1-\alpha)\left\|\begin{bmatrix}\mathbf{0}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}-\begin{bmatrix}\mathbf{0}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}-\rho\begin{bmatrix}\tilde{\mathscr{L}}\\ \mathbf{I}-\tilde{\mathscr{L}}\end{bmatrix}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^*\right)\right\|_{\boldsymbol{\theta}}^2,$$

which in turn means that the fixed point residual

$$\begin{aligned}&\left\|\begin{bmatrix}\mathbf{0}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}-\begin{bmatrix}\mathbf{0}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}-\rho\begin{bmatrix}\tilde{\mathscr{L}}\\ \mathbf{I}-\tilde{\mathscr{L}}\end{bmatrix}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^*\right)\right\|_{\boldsymbol{\theta}}^2 =\\ &\rho^2\left\|\tilde{\mathscr{L}}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^*\right)\right\|_{\tilde{\mathscr{L}}^\dagger}^2+\left\|\boldsymbol{\gamma}_R^{(k)}-\boldsymbol{\gamma}_R^{*}-\rho\left(\mathbf{I}-\tilde{\mathscr{L}}\right)\left(\mathbf{x}^{(k+1)}-\mathbf{x}^*\right)\right\|_{(\mathbf{I}-\tilde{\mathscr{L}})^\dagger}^2,\end{aligned} \tag{7.25}$$

converges to zero.

The convergence of (7.25) highlights two points. Firstly, as $\tilde{\mathscr{L}}\mathbf{x}^* = \mathbf{0}$ by definition, the convergence of $\tilde{\mathscr{L}}\left(\mathbf{x}^{(k)}-\mathbf{x}^*\right)$ ensures that $\tilde{\mathscr{L}}\mathbf{x}^{(k)} \to \mathbf{0}$ as well. As such, $\boldsymbol{\gamma}^{(k+1)}-\boldsymbol{\gamma}^{(k)} \to 0$ and furthermore $\langle \boldsymbol{\gamma}^{(k)},\mathbf{x}^{(k+1)}\rangle \to 0$. Similarly, as $\boldsymbol{\gamma}_R^*-\rho\left(\mathbf{I}-\tilde{\mathscr{L}}\right)\mathbf{x}^* = \mathbf{0}$, the convergence of $\boldsymbol{\gamma}_R^{(k)}-\boldsymbol{\gamma}_R^*-\rho\left(\mathbf{I}-\tilde{\mathscr{L}}\right)\left(\mathbf{x}^{(k+1)}-\mathbf{x}^*\right)$ ensures that $\boldsymbol{\gamma}_R^{(k)}-\rho\left(\mathbf{I}-\tilde{\mathscr{L}}\right)\mathbf{x}^{(k+1)} \to 0$. Combined with the fact that $\tilde{\mathscr{L}}\mathbf{x}^{(k)} \to \mathbf{0}$, it follows that $\boldsymbol{\gamma}_R^{(k)}-\rho\mathbf{x}^{(k+1)} \to 0$. Together with the primal update equations, these properties ensure that $\mathbf{x}^{(k+1)}-\mathbf{x}^{(k)} \to \mathbf{0}$ and thus that the primal variables converge to a fixed point. This in turn demonstrates that the $\boldsymbol{\gamma}$ variables converge to a fixed point which in turn guarantees optimality of the primal fixed points.

7.6. DISTRIBUTED TIME VARYING CONSENSUS

We now turn our attention to the original objective of this chapter; to demonstrate how the proposed topology dependent change of variables leads to an algorithm for use in time varying networks. As outlined in Section 7.3.1, for each time step k, we now assume that the network takes on a potentially different connected topology given by $G_{(k)}\left(V, E_{(k)}\right)$. The matrices $\mathbf{C}_{(k)}$ and $\mathbf{P}_{(k)}$, and subsequently $\mathbf{\Phi}_{(k)}$, $\mathscr{K}_{(k)}$, $\mathscr{R}_{(k)}$ and $\mathscr{L}_{(k)}$, are defined exactly as in the time invariant case for each $G_{(k)}$.

To form our proposed method for time varying consensus, we augment Algorithm 14 by allowing the weighted graph Laplacian $\tilde{\mathscr{L}}$ to also vary with time. Additionally, we will restrict ourselves to the case that $\alpha = \frac{1}{2}$. The particular choice of this averaging parameter can be motivated from the simplification it provides to the computation of the $\boldsymbol{\gamma_R}$ variables, notably that they become a linear function of the primal $\mathbf{x}$ variables. This can be observed in Algorithm 15, referred to as the time varying distributed consensus (TVDC) method in the remainder of the text, which inherits the appealing node based operation of its time invariant predecessor.

Algorithm 15 Time Varying Distributed Consensus (TVDC)

1: **Initialise:** $\boldsymbol{\gamma}_R^{(0)}, \boldsymbol{\gamma}_N^{(0)} = \mathbf{0}$
2: **for** $k = 0, \cdots,$ **do**
3: Using $G_{(k)}$ compute $\tilde{\mathscr{L}}_{(k)}$ as per Lemma 7.4.1
4: $\mathbf{x}^{(k+1)} = \underset{\mathbf{x}}{\operatorname{argmin}} \left(f(\mathbf{x}) - \left\langle \begin{bmatrix} \mathbf{I} \\ \mathbf{I} \end{bmatrix}^T \begin{bmatrix} \boldsymbol{\gamma}_K^{(k)} \\ \boldsymbol{\gamma}_R^{(k)} \end{bmatrix}, \mathbf{x} \right\rangle + \frac{\rho}{2} ||\mathbf{x}||^2 \right)$
5: $\begin{bmatrix} \boldsymbol{\gamma}_K^{(k+1)} \\ \boldsymbol{\gamma}_R^{(k+1)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\gamma}_K^{(k)} \\ \mathbf{0} \end{bmatrix} + \rho \begin{bmatrix} -\tilde{\mathscr{L}}_{(k)} \\ \mathbf{I} - \tilde{\mathscr{L}}_{(k)} \end{bmatrix} \mathbf{x}^{(k+1)}$
6: **end for**

In the following we demonstrate how this method provides a similar convergence guarantee to the time invariant case despite the change in the update equations at each iteration.

7.6.1. TVDC: TIME VARYING ALGORITHMIC CONVERGENCE

To prove the convergence of the proposed TVDC method we make two assumptions. Firstly we restrict the function f to be μ-strongly convex, which also implies that $\mathbf{x}^*$ is unique. As noted in Chapter 2, the strong convexity of a function is defined below.

Definition 7.6.1. *Strong Convexity: A function f is μ-strongly convex with $\mu > 0$ if $\forall \mathbf{x}_1, \mathbf{x}_2 \in dom(f), \mathbf{y}_2 \in \partial f(\mathbf{x}_2)$,*

$$f(\mathbf{x}_1) \geq f(\mathbf{x}_2) + \langle \mathbf{y}_2, \mathbf{x}_1 - \mathbf{x}_2 \rangle + \frac{\mu}{2} \|\mathbf{x}_1 - \mathbf{x}_2\|^2 .$$

Importantly, in this section we only demonstrate that these conditions are sufficient for convergence, not that they are necessary. In line with this statement, in Section 7.7 we demonstrate an example where these conditions do not hold while still allowing for convergence of the TVDC algorithm which suggests that a weaker convergence guarantee may be found. This has been left to future work.

To demonstrate convergence, we model the variation in network topology via a random process. Assume that for all iterations $k \in \mathbb{N}$, that the network topologies $G^{(k)}$ are drawn from an independent and identically distributed process. In other words, $\forall k, l \in \mathbb{N}$ and any graph $\mathcal{G}$, the probabilities $\text{prob}\left(G^{(k)} = \mathcal{G}\right) = \text{prob}\left(G^{(l)} = \mathcal{G}\right)$. With these assumptions in mind we can use statistical expectation as a method for proving convergence of Algorithm 15.

While we do not show that the TVDC algorithm will converge at every iteration, as was the case of the time invariant algorithm, using the results of the previous section we can show that $\mathbf{x}^{(k)} \to \mathbf{x}^*$ in expectation and variance where the point $\mathbf{x}^*$ is unique due to the strong convexity of f. Specifically, if $\mathbf{x}^{(k)}$ converges in expectation then

$$\left\| E_G\left[\mathbf{x}^{(k)} - \mathbf{x}^*\right]\right\|^2 \to 0,\ k \to \infty.$$

Here the notation E_G is used to denote expectation with respect to the random sequence of connected graphs. Similarly, if we have convergence in variance then

$$\text{tr}\left(E_G\left[\left(\mathbf{x}^{(k)} - \mathbf{x}^* - E_G\left[\mathbf{x}^{(k)} - \mathbf{x}^*\right]\right)\left(\mathbf{x}^{(k)} - \mathbf{x}^* - E_G\left[\mathbf{x}^{(k)} - \mathbf{x}^*\right]\right)^T\right]\right) \to 0,\ k \to \infty,$$

where $\text{tr}(\bullet)$ denotes the trace of a matrix. Fortunately, both conditions must hold by showing that $E_G\left[\left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\|^2\right] \to 0$. Convergence in expectation follows directly from Jensen's inequality which for this case states that

$$\left\| E_G\left[\mathbf{x}^{(k)} - \mathbf{x}^*\right]\right\|^2 \leq E_G\left[\left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\|^2\right].$$

Convergence in covariance can be shown by using a combination of the linearity of the trace and expectation operators to show that

$$\begin{aligned}
&\text{tr}\left(E_G\left[\left(\mathbf{x}^{(k)} - \mathbf{x}^* - E_G\left[\mathbf{x}^{(k)} - \mathbf{x}^*\right]\right)\left(\mathbf{x}^{(k)} - \mathbf{x}^* - E_G\left[\mathbf{x}^{(k)} - \mathbf{x}^*\right]\right)^T\right]\right) \\
&= \text{tr}\left(E_G\left[\left(\mathbf{x}^{(k)} - \mathbf{x}^*\right)\left(\mathbf{x}^{(k)} - \mathbf{x}^*\right)^T\right] - E_G\left[\mathbf{x}^{(k)} - \mathbf{x}^*\right] E_G\left[\mathbf{x}^{(k)} - \mathbf{x}^*\right]^T\right) \\
&= E_G\left[\left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\|^2\right] - \left\|E_G\left[\mathbf{x}^{(k)} - \mathbf{x}^*\right]\right\|^2 \leq E_G\left[\left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\|^2\right],
\end{aligned}$$

where in the final equality we have used the cyclic permutation property of the trace operator. It follows that if $E_G\left[\left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\|^2\right] \to 0$ then convergence is guaranteed in both expectation and variance as desired.

To show that $E_G\left[\left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\|^2\right] \to 0$, consider the sequence of expected norm terms

$$\left(E_G\left[\left\|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k+1)} \\ \boldsymbol{\gamma}_R^{(k+1)}\end{bmatrix} - \begin{bmatrix}\boldsymbol{\gamma}_K^* \\ \boldsymbol{\gamma}_R^*\end{bmatrix}\right\|_{\bar{\boldsymbol{\theta}}}^2\right]\right)_{k\in\mathbb{N}}, \tag{7.26}$$

where the matrix $\bar{\boldsymbol{\theta}}$ is given by

$$\bar{\boldsymbol{\theta}} = \begin{bmatrix} E_G\left[\tilde{\mathscr{L}}_{(k)}\right] & \mathbf{0} \\ \mathbf{0} & E_G\left[\left(\mathbf{I} - \tilde{\mathscr{L}}_{(k)}\right)\right] \end{bmatrix}^{\dagger}.$$

Specifically, we show that the sequence in (7.26) is nondecreasing and as a by-product, that $E_G\left[\left\|\mathbf{x}^{(k)}-\mathbf{x}^*\right\|^2\right]\to 0$ as desired. Like in the time invariant case, the quadratic form $\|\bullet\|_{\tilde{\boldsymbol{\theta}}}$ in (7.26) forms a valid metric for the given iterates as firstly all $\boldsymbol{\gamma}_K\in\operatorname{ran}\left(E_G\left[\tilde{\mathscr{L}}_{(k)}\right]\right)$ and secondly $\boldsymbol{\gamma}_R\in\operatorname{ran}\left(E_G\left[\mathbf{I}-\tilde{\mathscr{L}}_{(k)}\right]\right)$. The first property follows from a combination of the definition of the update equations for $\boldsymbol{\gamma}_K$ in Algorithm 15, the fact that any realization $\tilde{\mathscr{L}}_{(k)}\succeq\mathbf{0}$ (Lemma 7.4.3) and that for all realizations $\tilde{\mathscr{L}}_{(k)}$, the kernels $\ker\left(\tilde{\mathscr{L}}_{(k)}\right)$ are the same (Lemma 7.4.2). Due to its equivalence with a convex combination of realizations, the matrix $E_G\left[\tilde{\mathscr{L}}_{(k)}\right]$ therefore has the same range space as any single realization, which ensures that $\boldsymbol{\gamma}_K\in\operatorname{ran}\left(E_G\left[\tilde{\mathscr{L}}_{(k)}\right]\right)$ as desired.

The second property follows from the following Lemma.

Lemma 7.6.1. *For the set of variables $\left(\boldsymbol{\gamma}^{(k)}\right)_{k\in\mathbb{N}}$ generated as per Algorithm 15*

$$\boldsymbol{\gamma}_R^{(k)}\in\operatorname{ran}\left(E_G\left[\mathbf{I}-\tilde{\mathscr{L}}_{(k)}\right]\right)\ \forall k\in\mathbb{N}.$$

Proof. From Lemma 7.4.3, the eigenvalues of any realization $\tilde{\mathscr{L}}_{(k)}$ are contained within $[0,1]$. It follows that the eigenvalues of the matrix $\mathbf{I}-\tilde{\mathscr{L}}_{(k)}$ are also contained within this range. Therefore, as a convex combination of positive semidefinite matrices, the matrix $E_G\left[\mathbf{I}-\tilde{\mathscr{L}}_{(k)}\right]$ has a range space either equal to or larger than any one realization $\tilde{\mathscr{L}}_{(k)}$. To demonstrate this fact, consider the following scenario. Assume that for all possible realizations of $\tilde{\mathscr{L}}_{(k)}$ there exists a shared vector $\mathbf{v}$ so that $\tilde{\mathscr{L}}_{(k)}\mathbf{v}=\mathbf{v}$ for all such realizations. In this case, it follows that $E_G\left[\mathbf{I}-\tilde{\mathscr{L}}_{(k)}\right]$ would be rank deficient. However, from the update equations of $\boldsymbol{\gamma}_R$ in Algorithm 15, it follows that all $\boldsymbol{\gamma}_R\perp\mathbf{v}$ and thus $\boldsymbol{\gamma}_R\in\operatorname{ran}\left(E_G\left[\mathbf{I}-\tilde{\mathscr{L}}_{(k)}\right]\right)$. If such a $\mathbf{v}$ does not exist then $E_G\left[\mathbf{I}-\tilde{\mathscr{L}}_{(k)}\right]\succ\mathbf{0}$ and thus $\boldsymbol{\gamma}_R\in\operatorname{ran}\left(E_G\left[\mathbf{I}-\tilde{\mathscr{L}}_{(k)}\right]\right)=\mathbb{R}^{M_V}$ by definition. □

With these properties under our belt we are now ready to address the convergence of the sequence (7.26). To achieve this, we can use conditional expectation to rewrite the elements of the sequence as

$$E_G\left[\left\|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k+1)}\\ \boldsymbol{\gamma}_R^{(k+1)}\end{bmatrix}-\begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}\right\|_{\tilde{\boldsymbol{\theta}}}^2\right]=\sum_{\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}\in\boldsymbol{\Gamma}^{(k)}}\operatorname{prob}\left(\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}\right)E_G\left[\left\|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k+1)}\\ \boldsymbol{\gamma}_R^{(k+1)}\end{bmatrix}-\begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}\right\|_{\tilde{\boldsymbol{\theta}}}^2\middle|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}\right],\tag{7.27}$$

where $\boldsymbol{\Gamma}^{(k)}$ denotes the set of all possible $\boldsymbol{\gamma}$ vectors for the kth iteration. We can therefore approach the task of analysis of (7.26) by firstly considering each of the conditional expected norm terms

$$E_G\left[\left\|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k+1)}\\ \boldsymbol{\gamma}_R^{(k+1)}\end{bmatrix}-\begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}\right\|_{\tilde{\boldsymbol{\theta}}}^2\middle|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}\right]\ \forall\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}\in\boldsymbol{\Gamma}.$$

Specifically, by using the inequality given in (7.20) it follows that

$$E_G\left[\left\|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k+1)}\\ \boldsymbol{\gamma}_R^{(k+1)}\end{bmatrix}-\begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}\right\|_{\tilde{\boldsymbol{\theta}}}^2\,\middle|\,\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}\right]=\frac{1}{2}E_G\left[\left\|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}-\begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}-2\rho\begin{bmatrix}\tilde{\mathscr{L}}_{(k)}\\ \mathbf{I}-\tilde{\mathscr{L}}_{(k)}\end{bmatrix}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^{*}\right)\right\|_{\tilde{\boldsymbol{\theta}}}^2\,\middle|\,\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}\right]$$
$$+\frac{1}{2}\left\|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}-\begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}\right\|_{\tilde{\boldsymbol{\theta}}}^2-\frac{1}{4}E_G\left[\left\|\begin{bmatrix}\mathbf{0}\\ 2\boldsymbol{\gamma}_R^{(k)}\end{bmatrix}-\begin{bmatrix}\mathbf{0}\\ 2\boldsymbol{\gamma}_R^{*}\end{bmatrix}-2\rho\begin{bmatrix}\tilde{\mathscr{L}}_{(k)}\\ \mathbf{I}-\tilde{\mathscr{L}}_{(k)}\end{bmatrix}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^{*}\right)\right\|_{\tilde{\boldsymbol{\theta}}}^2\,\middle|\,\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}\right].$$

Therefore, if the inequality

$$\left\|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}-\begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}\right\|_{\tilde{\boldsymbol{\theta}}}^2\geq E_G\left[\left\|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}-\begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}-2\rho\begin{bmatrix}\tilde{\mathscr{L}}_{(k)}\\ \mathbf{I}-\tilde{\mathscr{L}}_{(k)}\end{bmatrix}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^{*}\right)\right\|_{\tilde{\boldsymbol{\theta}}}^2\,\middle|\,\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}\right],\tag{7.28}$$

holds for all $\boldsymbol{\gamma}^{(k)}\in\boldsymbol{\Gamma}^{(k)}$ then each conditional expected norm term will satisfy the inequality

$$E_G\left[\left\|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k+1)}\\ \boldsymbol{\gamma}_R^{(k+1)}\end{bmatrix}-\begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}\right\|_{\tilde{\boldsymbol{\theta}}}^2\,\middle|\,\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}\right]\leq\left\|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}-\begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}\right\|_{\tilde{\boldsymbol{\theta}}}^2,$$

and thus that, for any given $k\in\mathbb{N}$,

$$E_G\left[\left\|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k+1)}\\ \boldsymbol{\gamma}_R^{(k+1)}\end{bmatrix}-\begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}\right\|_{\tilde{\boldsymbol{\theta}}}^2\right]\leq E_G\left[\left\|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}-\begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}\right\|_{\tilde{\boldsymbol{\theta}}}^2\right].$$

To show that (7.28) holds for each conditional expected norm, we will again follow the procedure adopted in the time invariant case by noting that

$$E_G\left[\left\|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}-\begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}-2\rho\begin{bmatrix}\tilde{\mathscr{L}}_{(k)}\\ \mathbf{I}-\tilde{\mathscr{L}}_{(k)}\end{bmatrix}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^{*}\right)\right\|_{\tilde{\boldsymbol{\theta}}}^2\,\middle|\,\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}\right]=\left\|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}-\begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}\right\|_{\tilde{\boldsymbol{\theta}}}^2$$
$$+4E_G\left[\left\|\rho\begin{bmatrix}\tilde{\mathscr{L}}_{(k)}\\ \mathbf{I}-\tilde{\mathscr{L}}_{(k)}\end{bmatrix}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^{*}\right)\right\|_{\tilde{\boldsymbol{\theta}}}^2-\left\langle\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}-\begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix},\rho\begin{bmatrix}\tilde{\mathscr{L}}_{(k)}\\ \mathbf{I}-\tilde{\mathscr{L}}_{(k)}\end{bmatrix}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^{*}\right)\right\rangle_{\tilde{\boldsymbol{\theta}}}\,\middle|\,\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}\right].$$

It follows that a sufficient condition for (7.28) to hold is if

$$\begin{aligned}&E_G\left[\left\|\rho\begin{bmatrix}\tilde{\mathscr{L}}_{(k)}\\ \mathbf{I}-\tilde{\mathscr{L}}_{(k)}\end{bmatrix}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^{*}\right)\right\|_{\tilde{\boldsymbol{\theta}}}^2\,\middle|\,\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}\right]\\ &\leq E_G\left[\left\langle\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}-\begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix},\rho\begin{bmatrix}\tilde{\mathscr{L}}_{(k)}\\ \mathbf{I}-\tilde{\mathscr{L}}_{(k)}\end{bmatrix}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^{*}\right)\right\rangle_{\tilde{\boldsymbol{\theta}}}\,\middle|\,\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}\right]\\ &=\left\langle\begin{bmatrix}\mathbf{I}\\ \mathbf{I}\end{bmatrix}^{T}\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}-\begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix},\rho\left(\mathbf{x}^{(k+1)}-\mathbf{x}^{*}\right)\right\rangle.\end{aligned}\tag{7.29}$$

Note that the lack of expectation in the final equality stems from the fact that for a given realization of $\boldsymbol{\gamma}^k$, the primal $\mathbf{x}$ updates are deterministic (see Algorithm 15).

To show that (7.29) holds for our considered functional class, we make use of the inequality.

$$E_G\left[\left\|\begin{bmatrix}\tilde{\mathscr{L}}_{(k)}\\ \mathbf{I}-\tilde{\mathscr{L}}_{(k)}\end{bmatrix}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^*\right)\right\|_{\tilde{\boldsymbol{\theta}}}^2 \;\middle|\; \begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}\right] \leq \beta\left\|\mathbf{x}^{(k+1)}-\mathbf{x}^*\right\|^2,$$

where β corresponds to the largest eigenvalue of $E_G\left[\begin{bmatrix}\tilde{\mathscr{L}}\\ \mathbf{I}-\tilde{\mathscr{L}}\end{bmatrix}^T\bar{\boldsymbol{\theta}}\begin{bmatrix}\tilde{\mathscr{L}}\\ \mathbf{I}-\tilde{\mathscr{L}}\end{bmatrix}\right]$ so that

$$\beta=\lambda_{\max}\left(E_G\left[\begin{bmatrix}\tilde{\mathscr{L}}\\ \mathbf{I}-\tilde{\mathscr{L}}\end{bmatrix}^T\bar{\boldsymbol{\theta}}\begin{bmatrix}\tilde{\mathscr{L}}\\ \mathbf{I}-\tilde{\mathscr{L}}\end{bmatrix}\right]\right). \tag{7.30}$$

The following Lemma establishes the range of β.

Lemma 7.6.2. *For a given set of unique positive semidefinite matrices* $\{\mathbf{A}_i \mid \mathbf{I}\succeq\mathbf{A}_i\succeq\mathbf{0},\ i=1,\dots,I\}$ *and positive definite scalars* $\{\theta_i \mid 1>\theta_i>0,\ i=1,\dots,I\}$ *where* $\sum_i\theta_i=1$,

$$\lambda_{\max}\left(\sum_{i=1}^I\theta_i\begin{bmatrix}\mathbf{A}_i\\ \mathbf{I}-\mathbf{A}_i\end{bmatrix}^T\begin{bmatrix}\left(\sum_{i=1}^I\theta_i\mathbf{A}_i\right)^\dagger & \mathbf{0}\\ \mathbf{0} & \left(\sum_{i=1}^I\theta_i\left(\mathbf{I}-\mathbf{A}_i\right)\right)^\dagger\end{bmatrix}\begin{bmatrix}\mathbf{A}_i\\ \mathbf{I}-\mathbf{A}_i\end{bmatrix}\right)>1.$$

The proof this Lemma can be found in Appendix 7.F. In the case that the set of matrices correspond to the possible graph Laplacians $\tilde{\mathscr{L}}$ and the scalars θ_i correspond to their associated probabilities, this ensures that $\beta>1$. Using this fact, by imposing that $\rho\leq\frac{\mu}{\beta-1}$, it follows that

$$E_G\left[\left\|\rho\begin{bmatrix}\tilde{\mathscr{L}}_{(k)}\\ \mathbf{I}-\tilde{\mathscr{L}}_{(k)}\end{bmatrix}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^*\right)\right\|_{\tilde{\boldsymbol{\theta}}}^2 \;\middle|\; \begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}\right]=\rho^2E_G\left[\left\|\begin{bmatrix}\tilde{\mathscr{L}}_{(k)}\\ \mathbf{I}-\tilde{\mathscr{L}}_{(k)}\end{bmatrix}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^*\right)\right\|_{\tilde{\boldsymbol{\theta}}}^2 \;\middle|\; \begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}\right]$$
$$\leq\beta\rho^2\left\|\mathbf{x}^{(k+1)}-\mathbf{x}^*\right\|^2\leq\rho(\rho+\mu)\left\|\mathbf{x}^{(k+1)}-\mathbf{x}^*\right\|^2\leq\left\langle\begin{bmatrix}\mathbf{I}\\ \mathbf{I}\end{bmatrix}^T\left(\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}-\begin{bmatrix}\boldsymbol{\gamma}_K^*\\ \boldsymbol{\gamma}_R^*\end{bmatrix}\right),\rho\left(\mathbf{x}^{(k+1)}-\mathbf{x}^*\right)\right\rangle,$$

where the final inequality stems from the definition of the primal update equations and the $(\mu+\rho)$-strong convexity of $f(\bullet)+\frac{\rho}{2}\|\bullet\|^2$ which verifies that (7.29) holds. The conditional norm therefore satisfies the inequality

$$E_G\left[\left\|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k+1)}\\ \boldsymbol{\gamma}_R^{(k+1)}\end{bmatrix}-\begin{bmatrix}\boldsymbol{\gamma}_K^*\\ \boldsymbol{\gamma}_R^*\end{bmatrix}\right\|_{\tilde{\boldsymbol{\theta}}}^2 \;\middle|\; \begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}\right]\leq\left\|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}-\begin{bmatrix}\boldsymbol{\gamma}_K^*\\ \boldsymbol{\gamma}_R^*\end{bmatrix}\right\|_{\tilde{\boldsymbol{\theta}}}^2$$
$$-\frac{1}{4}E_G\left[\left\|\begin{bmatrix}\mathbf{0}\\ 2\boldsymbol{\gamma}_R^{(k)}\end{bmatrix}-\begin{bmatrix}\mathbf{0}\\ 2\boldsymbol{\gamma}_R^*\end{bmatrix}-2\rho\begin{bmatrix}\tilde{\mathscr{L}}_{(k)}\\ \mathbf{I}-\tilde{\mathscr{L}}_{(k)}\end{bmatrix}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^*\right)\right\|_{\tilde{\boldsymbol{\theta}}}^2 \;\middle|\; \begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}\right].$$

Substituting back into the original quadratic form in (7.27) we find that

$$E_G\left[\left\|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k+1)}\\ \boldsymbol{\gamma}_R^{(k+1)}\end{bmatrix}-\begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}\right\|_{\bar{\boldsymbol{\theta}}}^2\right] \leq E_G\left[\left\|\begin{bmatrix}\boldsymbol{\gamma}_K^{(k)}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}-\begin{bmatrix}\boldsymbol{\gamma}_K^{*}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}\right\|_{\bar{\boldsymbol{\theta}}}^2\right]$$
$$-E_G\left[\left\|\begin{bmatrix}\mathbf{0}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}-\begin{bmatrix}\mathbf{0}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}-\rho\begin{bmatrix}\tilde{\mathscr{L}}_{(k)}\\ \mathbf{I}-\tilde{\mathscr{L}}_{(k)}\end{bmatrix}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^{*}\right)\right\|_{\bar{\boldsymbol{\theta}}}^2\right],$$

where we have simplified the inequality by using the definition of expectation operation. From this we can immediately note that

$$E_G\left[\left\|\begin{bmatrix}\mathbf{0}\\ \boldsymbol{\gamma}_R^{(k)}\end{bmatrix}-\begin{bmatrix}\mathbf{0}\\ \boldsymbol{\gamma}_R^{*}\end{bmatrix}-\rho\begin{bmatrix}\tilde{\mathscr{L}}_{(k)}\\ \mathbf{I}-\tilde{\mathscr{L}}_{(k)}\end{bmatrix}\left(\mathbf{x}^{(k+1)}-\mathbf{x}^{*}\right)\right\|_{\bar{\boldsymbol{\theta}}}^2\right]\rightarrow 0,$$

which, like in Section 7.5, ensures convergence to a primal optimal fixed point. Due to the strong convexity assumption of f, this fixed point is unique which guarantees that $E_G\left[\left\|\mathbf{x}^{(k)}-\mathbf{x}^{*}\right\|^2\right]\rightarrow 0$ as desired and thus that the primal variables converge in both expectation and variance.

Remark 4. *It is important to highlight that the proof presented in this section does not explicitly rely on the choice of using a half averaged version of TVDC. In fact we only require that the algorithm be averaged for this proof to hold for the proposed functional class. The reason for restricting ourselves to the half averaged case is the empirical observation that this choice is also sufficient for convergence of more general functional classes as well. This point is demonstrated in the following section where we also show how, in practice, for strongly convex functions, larger step sizes can be used without causing algorithm divergence. For this reason, by restricting the averaging parameter to a factor of a half we postulate that the TVDC algorithm can be used for more general classes of functions and perhaps time varying network models as well, although no such proof exists for this at this time.*

7.7. SIMULATIONS

We now demonstrate the performance of this algorithm in solving two simple consensus problems. The first is the classic distributed averaging problem which satisfies the strong monotonicity assumptions for convergence while the second is a L1 version of the same problem. Interestingly, despite the fact that the second problem does not exhibit strong convexity, we can demonstrate convergence for a wide range of tested step sizes which alludes to the fact that a more general convergence proof may exist.

7.7.1. DISTRIBUTED AVERAGING

Consider the distributed averaging problem given by

$$\min_{\mathbf{x}\in\mathbb{R}^N}\sum_{i\in V}\frac{1}{2}\|\mathbf{x}_i-\mathbf{a}_i\|^2 \quad \text{s.t.}\,\mathbf{x}_i-\mathbf{x}_j=\mathbf{0}\;\forall(i,j)\in E,$$

where $\mathbf{a}_i$ are local observation vectors at each node and, by inspection, $\mathbf{x}_i^{*}=\frac{1}{N}\sum_{j\in V}\mathbf{a}_j\;\forall\, i\in V$. Note that f is 1-strongly convex in this case such that we can select any $\rho\in(0,\frac{1}{\beta-1}]$,

where in this case $\beta = 1.27$ as defined in (7.30) while still guaranteeing convergence. Such a problem is therefore a perfect candidate for testing the proposed TVDC algorithm.

For these simulations, we considered a 100 node network where each local variable was a single scalar given by $x_i \in \mathbb{R}$. Furthermore, each node was equipped with a local observation $a_i \in \mathbb{R}$. Echoing our approach in previous chapters, the realizations of the time varying networks for each iteration were generated via an Erdős-Rényi model with a connection probability of $\frac{\log(N)}{N}$. The resulting networks were verified as forming a single connected network, as per our initial assumptions. The convergence curves for the resulting simulations are presented in Figure 7.2 where the final accuracy stems from the physical hardware utilized. Due to the uniqueness of the optimal primal variables $\mathbf{x}^*$, in this instance we use the quadratic form $\|\mathbf{x}-\mathbf{x}^*\|^2$ as a measure of convergence.

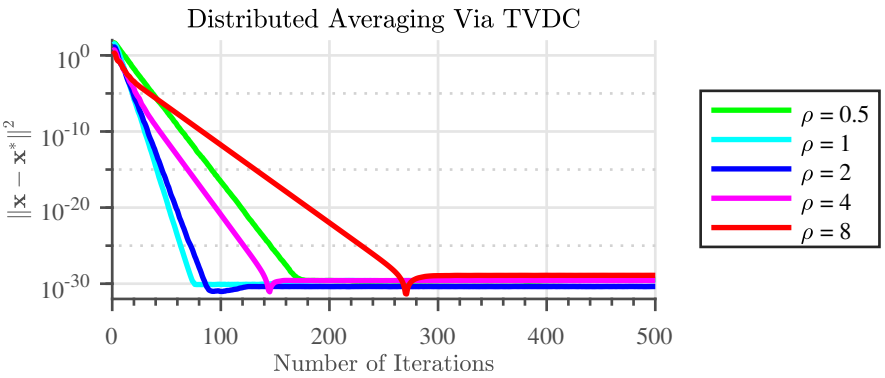


Figure 7.2: The primal objective gap of TVDC in solving the distributed averaging problem in a 100 node network for a range of step sizes.

It is interesting to note that despite the fact that f is only 1-strongly convex, we can use much larger step sizes than $\rho = \frac{1}{\beta-1}$, which is ≈ 3.7 in this case, while still achieving convergence. Furthermore, we note that if the strong convexity parameter is not known exactly, choosing a smaller step size to ensure that the conditions of the convergence proof are met can result in a degradation in convergence rate.

7.7.2. DISTRIBUTED L1 CONSENSUS

In the distributed averaging case, we observed that larger step sizes than those suggested by the convergence proof still resulted in algorithmic convergence. Motivated by this point, in the following simulation we demonstrate a simple L1 problem example for which the proposed TVDC algorithm converges despite the fact that the problem exhibits no strong convexity. The aforementioned problem is a simple variant of the distributed averaging problem with the quadratic form replaced with an L1 norm term. The resulting problem is given by

$$\min_{\mathbf{x}\in\mathbb{R}^N} \sum_{i\in V} \|\mathbf{x}_i - \mathbf{a}_i\|_1 \quad \text{s.t.}\, \mathbf{x}_i - \mathbf{x}_j = \mathbf{0}\ \forall (i,j) \in E$$

As in the previous subsection, we considered solving this problem for the case of a 100 node network where each node was equipped with a single observation and local scalar variable. We additionally adopted the same Erdős-Rényi model for the network realizations at each iteration and ensured that at each iteration the network formed a single connected component. The convergence curves for the resulting simulations are presented in Figure 7.3. Unlike the previous instance, as here the primal minimizers need not be unique, we instead demonstrate convergence of the primal objective function to its optimal configuration.

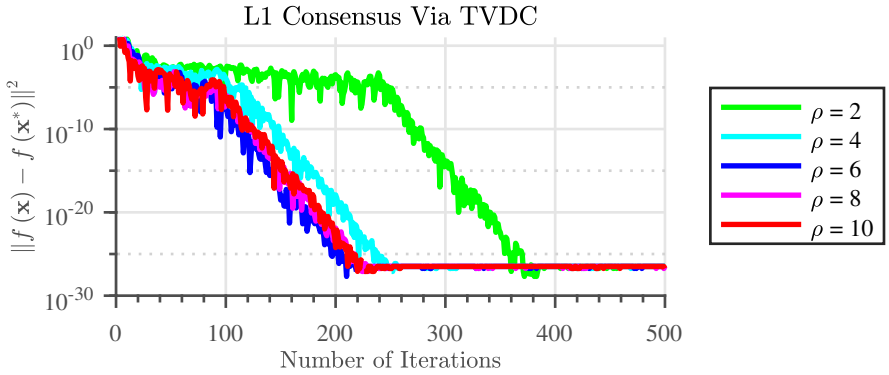


Figure 7.3: The primal objective gap of TVDC in solving an L1 distributed consensus problem in a 100 node network for a range of step sizes.

As in the case of its strongly convex predecessor, the L1 consensus problem poses no problem for the TVDC algorithm in this instance. While the convergence rate of the variables is still dependent on the step size selected, the algorithm converges for all ρ considered. Combined with the observation that larger step sizes could be used in the case of the distributed averaging problem, this result suggests that there may exist a more general convergence proof for the TVDC algorithm which allows it to be used for the more general case of CCP type objective functions. Contrasting with existing approaches in the literature which typically rely on strict assumptions of the function class, such a general purpose solver for time varying networks would be highly appealing.

7.8. CONCLUSION

In this chapter we have introduced a new algorithm for distributed consensus in time-varying networks. Based on a combination of averaged Peaceman-Rachford splitting and a particular network dependent change of variables, the proposed time varying distributed consensus (TVDC) algorithm was shown to have guaranteed convergence, a point which was demonstrated for two simple example consensus problems. Furthermore, TVDC was shown to exhibit a node based form with the algorithm itself being a combination of local optimization problems at each node and a particular weighted Laplacian mixing stage. This type of structure reflects similar efforts within the literature while having the distinct benefit of being derived from monotone operator theory. We

demonstrated how the node based structure of TVDC to be used to solve any strongly convex problems including those which are non-differentiable while retaining the same convergence guarantees. The convergence of standard closed, convex and proper problems was also considered and demonstrated empirically for a distributed L1 example. Ultimately, the proposed method provides an efficient approach for solving distributed consensus problems in time varying networks.

APPENDICES

7.A. PROOF OF LEMMA 7.3.1

As $\mathbf{R}_{\mathbf{T}_1,\rho} = 2\mathbf{J}_{\mathbf{T}_1,\rho} - \mathbf{I}$, we begin by defining a method for computing the update

$$\tilde{\boldsymbol{\lambda}} = \mathbf{J}_{\mathbf{T}_1,\rho}(\mathbf{z}).$$

Firstly, by the definition of the resolvent of an operator,

$$\tilde{\boldsymbol{\lambda}} = \left(\mathbf{I} + \rho\mathbf{T}_1\right)^{-1}(\mathbf{z}) \iff \mathbf{z} \in \left(\mathbf{I} + \rho\mathbf{T}_1\right)\left(\tilde{\boldsymbol{\lambda}}\right) \iff \tilde{\boldsymbol{\lambda}} \in \mathbf{z} - \rho\mathbf{T}_1\left(\tilde{\boldsymbol{\lambda}}\right).$$

From the definition of the operator $\mathbf{T}_1$, it follows that

$$\tilde{\boldsymbol{\lambda}} \in \mathbf{z} - \rho\left(\boldsymbol{\Phi}\mathbf{C}\partial f^*\left(\mathbf{C}^T\boldsymbol{\Phi}\tilde{\boldsymbol{\lambda}}\right)\right).$$

Let $\mathbf{x} \in \partial f^*\left(\mathbf{C}^T\boldsymbol{\Phi}\tilde{\boldsymbol{\lambda}}\right)$. For $f \in \Gamma_0$, it follows that $\mathbf{x} \in \partial f^*\left(\mathbf{C}^T\boldsymbol{\Phi}\tilde{\boldsymbol{\lambda}}\right) \iff \partial f(\mathbf{x}) \ni \mathbf{C}^T\boldsymbol{\Phi}\tilde{\boldsymbol{\lambda}}$ so that

$$\tilde{\boldsymbol{\lambda}} = \mathbf{z} - \rho\boldsymbol{\Phi}\mathbf{C}\mathbf{x},\ \mathbf{C}^T\boldsymbol{\Phi}\tilde{\boldsymbol{\lambda}} \in \partial f(\mathbf{x}). \tag{7.31}$$

Thus, $\mathbf{x}$ can be computed as

$$\mathbf{0} \in \partial f(\mathbf{x}) - \mathbf{C}^T\boldsymbol{\Phi}\left(\mathbf{z} - \rho\boldsymbol{\Phi}\mathbf{C}\mathbf{x}\right),$$

or equivalently as

$$\mathbf{x} = \arg\min_{\mathbf{x}}\left(f(\mathbf{x}) - \left\langle\mathbf{C}^T\boldsymbol{\Phi}\mathbf{z},\mathbf{x}\right\rangle + \frac{\rho}{2}||\boldsymbol{\Phi}\mathbf{C}\mathbf{x}||^2\right). \tag{7.32}$$

Combining (7.31) and (7.32) with the fact that $\mathbf{y} = \left(2\mathbf{J}_{\mathbf{T}_1,\rho} - \mathbf{I}\right)(\mathbf{z}) = 2\tilde{\boldsymbol{\lambda}} - \mathbf{z}$ completes the proof.

□

7.B. PROOF OF LEMMA 7.3.2

As in the proof Lemma 7.3.1 (see Appendix 7.A), we begin by utilising the definition of the reflected resolvent where $\mathbf{R}_{\mathbf{T}_2,\rho} = 2\mathbf{J}_{\mathbf{T}_2,\rho} - \mathbf{I}$. We therefore shift our attention to the computation of $\mathbf{J}_{\mathbf{T}_2,\rho}$.

The resolvent of a normal cone operator can be computed via

$$\mathbf{J}_{\mathbf{T}_2,\rho}(\mathbf{y}) = \arg\min_{\mathbf{u}}\left(\iota_{\ker((\mathbf{I}+\mathbf{P})\boldsymbol{\Phi})}(\mathbf{u}) + \frac{1}{2\rho}\|\mathbf{u}-\mathbf{y}\|^2\right) = \arg\min_{\boldsymbol{\Phi}\mathbf{u}+\mathbf{P}\boldsymbol{\Phi}\mathbf{u}=\mathbf{0}}\left(\|\mathbf{u}-\mathbf{y}\|^2\right).$$

By inspection, the solution of this problem is given by the projection onto the set of feasible $\mathbf{u}$ vectors. As such,

$$\mathbf{J}_{\mathbf{T}_2,\rho}(\mathbf{y}) = \prod_{\ker((\mathbf{I}+\mathbf{P})\mathbf{\Phi})} \mathbf{y} = \mathscr{K}\mathbf{y}.$$

It follows that the reflected resolvent can be computed as

$$\mathbf{R}_{\mathbf{T}_2,\rho}(\mathbf{y}) = \left(2\prod_{\ker((\mathbf{I}+\mathbf{P})\mathbf{\Phi})} - \mathbf{I}\right)\mathbf{y} = (\mathscr{K}-\mathscr{R})\,\mathbf{y},$$

completing the proof.

□

7.C. PROOF OF LEMMA 7.4.1

To derive the compact form of $\tilde{\mathscr{L}} = \mathbf{C}^T\mathbf{\Phi}\mathscr{K}\mathbf{\Phi}\mathbf{C}$, we will firstly consider the matrix

$$\mathbf{B} = \mathbf{\Phi}\mathscr{K}\mathbf{\Phi} = \mathbf{\Phi}\mathbf{\Phi} - \mathbf{\Phi}\mathscr{R}\mathbf{\Phi}.$$

Expanding the definition of $\mathscr{R}$, it follows that

$$\mathbf{B} = \mathbf{\Phi}\prod_{\ker((\mathbf{I}+\mathbf{P})\mathbf{\Phi})}\mathbf{\Phi} = \mathbf{\Phi}\mathbf{\Phi} - \mathbf{\Phi}\mathbf{\Phi}(\mathbf{I}+\mathbf{P})\left((\mathbf{I}+\mathbf{P})\mathbf{\Phi}\mathbf{\Phi}(\mathbf{I}+\mathbf{P})\right)^{\dagger}(\mathbf{I}+\mathbf{P})\mathbf{\Phi}\mathbf{\Phi}.$$

Given the diagonal nature of $\mathbf{\Phi}$, it follows that $\mathbf{B}$ is symmetrically permutable to a block diagonal form with each block having a size of just two by two. Each such block corresponds to an edge (i,j) and is given by

$$\mathbf{B}_{i,j} = \begin{bmatrix}\frac{1}{|\mathscr{N}(i)|} & 0\\ 0 & \frac{1}{|\mathscr{N}(j)|}\end{bmatrix} - \begin{bmatrix}\frac{1}{|\mathscr{N}(i)|} & 0\\ 0 & \frac{1}{|\mathscr{N}(j)|}\end{bmatrix}\begin{bmatrix}1&1\\1&1\end{bmatrix}\left(\begin{bmatrix}1&1\\1&1\end{bmatrix}\begin{bmatrix}\frac{1}{|\mathscr{N}(i)|} & 0\\ 0 & \frac{1}{|\mathscr{N}(j)|}\end{bmatrix}\begin{bmatrix}1&1\\1&1\end{bmatrix}\right)^{\dagger}\begin{bmatrix}1&1\\1&1\end{bmatrix}\begin{bmatrix}\frac{1}{|\mathscr{N}(i)|} & 0\\ 0 & \frac{1}{|\mathscr{N}(j)|}\end{bmatrix}.$$

With some manipulation, this can be rewritten as

$$\begin{aligned}\mathbf{B}_{i,j} &= \begin{bmatrix}\frac{1}{|\mathscr{N}(i)|} & 0\\ 0 & \frac{1}{|\mathscr{N}(j)|}\end{bmatrix} - \frac{\begin{bmatrix}\frac{1}{|\mathscr{N}(i)|} & 0\\ 0 & \frac{1}{|\mathscr{N}(j)|}\end{bmatrix}\begin{bmatrix}1&1\\1&1\end{bmatrix}\begin{bmatrix}\frac{1}{|\mathscr{N}(i)|} & 0\\ 0 & \frac{1}{|\mathscr{N}(j)|}\end{bmatrix}}{\frac{1}{|\mathscr{N}(i)|}+\frac{1}{|\mathscr{N}(j)|}}\\ &= \frac{\begin{bmatrix}1+\frac{|\mathscr{N}(j)|}{|\mathscr{N}(i)|} & 0\\ 0 & 1+\frac{|\mathscr{N}(i)|}{|\mathscr{N}(j)|}\end{bmatrix}}{|\mathscr{N}(i)|+|\mathscr{N}(j)|} - \frac{\begin{bmatrix}\frac{|\mathscr{N}(j)|}{|\mathscr{N}(i)|} & 1\\ 1 & \frac{|\mathscr{N}(i)|}{|\mathscr{N}(j)|}\end{bmatrix}}{|\mathscr{N}(i)|+|\mathscr{N}(j)|} = \frac{\begin{bmatrix}1&-1\\-1&1\end{bmatrix}}{|\mathscr{N}(i)|+|\mathscr{N}(j)|}\end{aligned}$$

As the matrix $\mathbf{C}$ simply causes a summation of edge based terms at each node, it follows that $\tilde{\mathscr{L}} = \mathbf{C}^T\mathbf{B}\mathbf{C} = \hat{\mathscr{L}}\otimes\mathbf{I}_M$ where $\otimes$ denotes the Kronecker product and the matrix $\hat{\mathscr{L}} \in \mathbb{R}^{N\times N}$ is given by

$$\left[\hat{\mathscr{L}}\right]_{i,j} = \begin{cases}\frac{-1}{|\mathscr{N}(i)|+|\mathscr{N}(j)|} & \text{if } (i,j)\in E\\ \sum_{j\in\mathscr{N}(i)}\frac{1}{|\mathscr{N}(i)|+|\mathscr{N}(j)|} & \text{if } i=j\\ 0 & \text{otherwise}\end{cases}.$$

The matrix $\hat{\mathscr{L}} = \hat{\mathbf{D}} - \hat{\mathbf{W}}$ can be thought of as a weighted graph Laplacian with weighted adjacency matrix $\hat{\mathbf{W}}$ and weighted degree matrix $\hat{\mathbf{D}} = \text{diag}\left(\hat{\mathbf{W}}\mathbf{1}_N\right)$. This matrix describes a process of weighted data combination at each node in the network.

Furthermore, we can note that $\hat{\mathbf{W}}$ can be computed as

$$\hat{\mathbf{W}} = \mathbf{W} \oslash \left(\mathbf{W}\mathbf{1}_N\mathbf{1}_N^T + \mathbf{1}_N\mathbf{1}_N^T\mathbf{W}\right),$$

where $\oslash$ denotes the element-wise division of a matrix. Using this information, we can define the matrix $\tilde{\mathscr{L}}$ so that

$$\tilde{\mathscr{L}} = \tilde{\mathbf{W}} - \tilde{\mathbf{D}} = \hat{\mathscr{L}} \otimes \mathbf{I}_M,$$

which is also a weighted graph Laplacian with

$$\tilde{\mathbf{W}} = \hat{\mathbf{W}} \otimes \mathbf{I}_M,\ \tilde{\mathbf{D}} = \text{diag}\left(\tilde{\mathbf{W}}\mathbf{1}_{NM}\right) = \text{diag}\left(\hat{\mathbf{W}}\mathbf{1}_N\right) \otimes \mathbf{I}_M.$$

This completes the proof.

□

7.D. PROOF OF LEMMA 7.4.2

Firstly, note that due to the particular structure of $\tilde{\mathscr{L}} = \hat{\mathscr{L}} \otimes \mathbf{I}_M$, that its eigenvalue decomposition $\tilde{\mathscr{L}} = \tilde{\mathbf{U}}\tilde{\mathbf{\Lambda}}\tilde{\mathbf{U}}^T$ can be equivalently written as based on the eigenvalue decomposition $\hat{\mathscr{L}} = \hat{\mathbf{U}}\hat{\mathbf{\Lambda}}\hat{\mathbf{U}}^T$. Specifically, it can be shown that

$$\tilde{\mathscr{L}} = \hat{\mathscr{L}} \otimes \mathbf{I}_M = \left(\hat{\mathbf{U}}\hat{\mathbf{\Lambda}}\hat{\mathbf{U}}^T\right) \otimes \mathbf{I}_M = \left(\hat{\mathbf{U}} \otimes \mathbf{I}_M\right)\left(\hat{\mathbf{\Lambda}} \otimes \mathbf{I}_M\right)\left(\hat{\mathbf{U}}^T \otimes \mathbf{I}_M\right) = \tilde{\mathbf{U}}\tilde{\mathbf{\Lambda}}\tilde{\mathbf{U}}^T, \tag{7.33}$$

where in the third equality we have twice used the mixed product property of Kronecker products. By defining the kernel of $\hat{\mathscr{L}}$ we can therefore derive that of $\tilde{\mathscr{L}}$. Considering this point, for 0 to be an eigenvalue of $\hat{\mathscr{L}}$, the corresponding eigenvector $\mathbf{x}$ must satisfy the equality

$$\mathbf{x}^T\hat{\mathscr{L}}\mathbf{x} = \mathbf{x}^T\left(\hat{\mathbf{D}} - \hat{\mathbf{W}}\right)\mathbf{x} = 0.$$

By rewriting the left hand term and using the definitions of $\hat{\mathbf{D}}$ and $\hat{\mathbf{W}}$, it follows that

$$\sum_{i\in V}\sum_{j\in V} x_i[\hat{\mathbf{W}}]_{i,j}(x_i - x_j) = \sum_{i\in V}\sum_{j\in V}\frac{x_i(x_i - x_j)}{|\mathscr{N}(i)| + |\mathscr{N}(j)|} = 0. \tag{7.34}$$

Using the symmetry of $\hat{\mathbf{W}}$, (7.34) can be equivalently written as

$$\sum_{(i,j)\in E}\frac{(x_i - x_j)^2}{|\mathscr{N}(i)| + |\mathscr{N}(j)|} = 0.$$

As $|\mathscr{N}(i)| \geq 1$ for all i, it follows that $x_i = x_j\ \forall (i,j) \in E$. As our networks are assumed connected, it follows that $x_i = \frac{1}{\sqrt{N}}\ \forall i \in V$ which is a single unique vector. As such $\hat{\mathbf{U}} = \frac{1}{\sqrt{N}}\mathbf{1}_N$. Combined with the definitions of $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{\Lambda}}$ in (7.33) completes the proof.

□

7.E. PROOF OF LEMMA 7.4.3

As in the case of the proof of Lemma 7.4.2 in Appendix 7.D, we can use the relationship between the eigenvalue decompositions of $\tilde{\mathscr{L}}$ and $\hat{\mathscr{L}}$ to complete this proof. Specifically, by bounding the spectra of $\hat{\mathscr{L}}$, using (7.33), we can equivalently bound that of $\tilde{\mathscr{L}}$ in turn. With this point in mind, we begin by first noting that $\hat{\mathscr{L}}$ is diagonally dominant and thus is positive semidefinite such that its eigenvalues are lower bounded by 0. The same therefore holds for $\tilde{\mathscr{L}}$.

To upper bound the eigenvalues of $\hat{\mathscr{L}}$ to be less than 1 we can equivalently show that its Rayleigh quotient satisfies

$$\frac{\mathbf{x}^T\hat{\mathscr{L}}\mathbf{x}}{\mathbf{x}^T\mathbf{x}} \leq 1 \ \forall \mathbf{x}. \tag{7.35}$$

By rearranging the Rayleigh quotient inequality above, it follows that we can equivalently show that

$$\mathbf{x}^T\hat{\mathscr{L}}\mathbf{x} - \mathbf{x}^T\mathbf{x} \leq 0 \ \forall \mathbf{x}. \tag{7.36}$$

To demonstrate this point, firstly note that we can rewrite the left hand term of (7.36) as

$$\begin{aligned}\mathbf{x}^T\hat{\mathscr{L}}\mathbf{x} &= \mathbf{x}^T\left(\tilde{\mathbf{D}} - \tilde{\mathbf{W}}\right)\mathbf{x} = \sum_{i\in V} x_i\left(\sum_{j\in\mathscr{N}(i)}\left[\tilde{\mathbf{W}}\right]_i^T\mathbf{1}x_i - \left[\tilde{\mathbf{W}}\right]_{i,j}x_j\right)\\ &= \sum_{i\in V}\sum_{j\in\mathscr{N}(i)}\frac{x_i^2 - x_ix_j}{|\mathscr{N}(i)|+|\mathscr{N}(j)|} = \sum_{(i,j)\in E}\frac{\left(x_i-x_j\right)^2}{|\mathscr{N}(i)|+|\mathscr{N}(j)|}.\end{aligned} \tag{7.37}$$

It follows that (7.37) can be equivalently written as

$$\begin{aligned}\mathbf{x}^T\hat{\mathscr{L}}\mathbf{x} - \mathbf{x}^T\mathbf{x} &= \sum_{(i,j)\in E}\frac{\left(x_i-x_j\right)^2}{|\mathscr{N}(i)|+|\mathscr{N}(j)|} - \sum_{i\in V}x_i^2\\ &= \sum_{(i,j)\in E}\left(\frac{\left(x_i-x_j\right)^2}{|\mathscr{N}(i)|+|\mathscr{N}(j)|} - \frac{x_i^2}{|\mathscr{N}(i)|} - \frac{x_j^2}{|\mathscr{N}(j)|}\right)\\ &= \sum_{(i,j)\in E}\frac{-|\mathscr{N}(j)|^2x_i^2 - 2|\mathscr{N}(i)||\mathscr{N}(j)|x_ix_j - |\mathscr{N}(i)|^2x_j^2}{|\mathscr{N}(i)|\left(|\mathscr{N}(i)|+|\mathscr{N}(j)|\right)|\mathscr{N}(j)|}\\ &= \sum_{(i,j)\in E}\frac{-\left(|\mathscr{N}(j)|x_i + |\mathscr{N}(i)|x_j\right)^2}{|\mathscr{N}(i)|\left(|\mathscr{N}(i)|+|\mathscr{N}(j)|\right)|\mathscr{N}(j)|} \leq 0.\end{aligned}$$

It follows that (7.35) holds and thus that the eigenvalues of $\hat{\mathscr{L}}$ are contained within the set $[0,1]$.

We can also note from (7.35) that for 1 to be an eigenvalue of $\hat{\mathscr{L}}$, then $|\mathscr{N}(j)|x_i = -|\mathscr{N}(i)|x_j \ \forall (i,j)\in E$. This can only be true if the underlying network is bipartite, i.e., that V can be partitioned into two disjoint sets V_1 and V_2 so that $\forall (i,j)\in E$, $i\in V_1, j\in V_2$. Any other network topology cannot satisfy this point and thus the upper bound must hold with strict inequality for all but bipartite graphs.

□

7.F. PROOF OF LEMMA 7.6.2

Given set of unique positive semidefinite matrices $\{\mathbf{A}_i \mid \mathbf{I} \succeq \mathbf{A}_i \succeq \mathbf{0},\ i = 1,\dots,I\}$ and positive definite scalars $\{\theta_i \mid 1 > \theta_i > 0,\ i = 1,\dots,I\}$ where $\sum_{i=1}^{I} \theta_i = 1$, to prove that

$$\lambda_{\max}\left(\sum_{i=1}^{I}\theta_i\begin{bmatrix}\mathbf{A}_i\\ \mathbf{I}-\mathbf{A}_i\end{bmatrix}^T\begin{bmatrix}\left(\sum_{i=1}^{I}\theta_i\mathbf{A}_i\right)^\dagger & \mathbf{0}\\ \mathbf{0} & \left(\sum_{i=1}^{I}\theta_i\left(\mathbf{I}-\mathbf{A}_i\right)\right)^\dagger\end{bmatrix}\begin{bmatrix}\mathbf{A}_i\\ \mathbf{I}-\mathbf{A}_i\end{bmatrix}\right) > 1,$$

we shall demonstrate that the matrix

$$\sum_{i=1}^{I}\theta_i\begin{bmatrix}\mathbf{A}_i\\ \mathbf{I}-\mathbf{A}_i\end{bmatrix}^T\begin{bmatrix}\left(\sum_{i=1}^{I}\theta_i\mathbf{A}_i\right)^\dagger & \mathbf{0}\\ \mathbf{0} & \left(\sum_{i=1}^{I}\theta_i\left(\mathbf{I}-\mathbf{A}_i\right)\right)^\dagger\end{bmatrix}\begin{bmatrix}\mathbf{A}_i\\ \mathbf{I}-\mathbf{A}_i\end{bmatrix}, \tag{7.38}$$

is equivalent to an identity matrix plus a summation of positive semidefinite matrices. To do so, we define the additional matrix $\mathbf{C} = \sum_{i=1}^{I}\theta_i\mathbf{A}_i$. As a convex combination of $\mathbf{A}_i$'s, $\mathbf{C}$ also satisfies the matrix inequality $\mathbf{I} \succeq \mathbf{C} \succeq \mathbf{0}$ such that $\mathbf{I} \succeq \mathbf{I}-\mathbf{C} \succeq \mathbf{0}$. Using this notation, we begin by noting that the matrix in (7.38) can be equivalently written as

$$\sum_{i=1}^{I}\theta_i\left(\mathbf{A}_i\mathbf{C}^\dagger\mathbf{A}_i + \left(\mathbf{I}-\mathbf{A}_i\right)\left(\mathbf{I}-\mathbf{C}\right)^\dagger\left(\mathbf{I}-\mathbf{A}_i\right)\right). \tag{7.39}$$

By expanding right hand term, (7.39) can be equivalently written as

$$\sum_{i=1}^{I}\theta_i\left(\mathbf{A}_i\left(\mathbf{C}^\dagger + \left(\mathbf{I}-\mathbf{C}\right)^\dagger\right)\mathbf{A}_i - \mathbf{A}_i\left(\mathbf{I}-\mathbf{C}\right)^\dagger - \left(\mathbf{I}-\mathbf{C}\right)^\dagger\mathbf{A}_i + \left(\mathbf{I}-\mathbf{C}\right)^\dagger\right), \tag{7.40}$$

which, using the definition of $\mathbf{C}$, can be more compactly be written as

$$\sum_{i=1}^{I}\theta_i\mathbf{A}_i\left(\mathbf{C}^\dagger + \left(\mathbf{I}-\mathbf{C}\right)^\dagger\right)\mathbf{A}_i + \left(\mathbf{I}-\mathbf{C}\right)\left(\mathbf{I}-\mathbf{C}\right)^\dagger + \left(\mathbf{I}-\mathbf{C}\right)^\dagger\left(\mathbf{I}-\mathbf{C}\right) - \left(\mathbf{I}-\mathbf{C}\right)^\dagger.$$

We can then define a family of equivalent matrices, one for each index i, by adding zero in the form of $(1-1)\sum_{j=1,j\neq i}^{I}\left(\mathbf{A}_j\left(\mathbf{C}^\dagger + \left(\mathbf{I}-\mathbf{C}\right)^\dagger\right)\mathbf{A}_i\right)$, so that (7.40) is equivalent to

$$\begin{aligned}\left(\mathbf{C}\mathbf{C}^\dagger + \mathbf{C}\left(\mathbf{I}-\mathbf{C}\right)^\dagger\right)\mathbf{A}_i - \sum_{j=1,j\neq i}\theta_j\mathbf{A}_j\left(\mathbf{C}^\dagger + \left(\mathbf{I}-\mathbf{C}\right)^\dagger\right)\left(\mathbf{A}_i-\mathbf{A}_j\right)\\ + \left(\mathbf{I}-\mathbf{C}\right)\left(\mathbf{I}-\mathbf{C}\right)^\dagger + \left(\mathbf{I}-\mathbf{C}\right)^\dagger\left(\mathbf{I}-\mathbf{C}\right) - \left(\mathbf{I}-\mathbf{C}\right)^\dagger.\end{aligned} \tag{7.41}$$

Taking convex combinations of (7.41) for all i where each matrix is weighted by its corresponding θ_i it follows that (7.40) can be rewritten as

$$\begin{aligned}\mathbf{C}\mathbf{C}^\dagger\mathbf{C} + \mathbf{C}\left(\mathbf{I}-\mathbf{C}\right)^\dagger\mathbf{C} + \left(\mathbf{I}-\mathbf{C}\right)\left(\mathbf{I}-\mathbf{C}\right)^\dagger + \left(\mathbf{I}-\mathbf{C}\right)^\dagger\left(\mathbf{I}-\mathbf{C}\right) - \left(\mathbf{I}-\mathbf{C}\right)^\dagger\\ + \sum_{i=1}^{I}\sum_{j=1}^{I}\frac{\theta_i\theta_j}{2}\left(\mathbf{A}_i-\mathbf{A}_j\right)\left(\left(\mathbf{I}-\mathbf{C}\right)^\dagger + \mathbf{C}^\dagger\right)\left(\mathbf{A}_i-\mathbf{A}_j\right).\end{aligned} \tag{7.42}$$

By then noting that $(\mathbf{I}-\mathbf{C})(\mathbf{I}-\mathbf{C})^{\dagger}(\mathbf{I}-\mathbf{C}) = \mathbf{C}(\mathbf{I}-\mathbf{C})^{\dagger}\mathbf{C} + (\mathbf{I}-\mathbf{C})(\mathbf{I}-\mathbf{C})^{\dagger} + (\mathbf{I}-\mathbf{C})^{\dagger}(\mathbf{I}-\mathbf{C}) - (\mathbf{I}-\mathbf{C})$ and using the fact that $\mathbf{C}\mathbf{C}^{\dagger}\mathbf{C} = \mathbf{C}$ and $(\mathbf{I}-\mathbf{C})(\mathbf{I}-\mathbf{C})^{\dagger}(\mathbf{I}-\mathbf{C}) = (\mathbf{I}-\mathbf{C})$, it follows that we can compactly write (7.42) as

$$\mathbf{I} + \sum_{i=1}^{I}\sum_{j=1}^{I}\frac{\theta_i\theta_j}{2}\left(\mathbf{A}_i - \mathbf{A}_j\right)\left((\mathbf{I}-\mathbf{C})^{\dagger} + \mathbf{C}^{\dagger}\right)\left(\mathbf{A}_i - \mathbf{A}_j\right). \tag{7.43}$$

As (7.43) is the sum of an identity matrix and a sum of positive definite quadratic forms, it follows that the largest eigenvalue of (7.39) must be strictly larger than one, completing the proof.

□

IV

Practical Distributed Convex Optimization

8

Robust Distributed Linearly Constrained Beamforming

Andreas I. Koutrouvelis, Thomas Sherson, Richard Heusdens and Richard C. Hendriks

"Theory without practice is empty. Practice without theory is blind."

Immanuel Kant

This chapter is based on the article published as "A Low-Cost Robust Distributed Linearly Constrained Beamformer for Wireless Acoustic Sensor Networks with Arbitrary Topology" by A.I. Koutrouvelis, T.W Sherson, R Heusdens and R.C. Hendriks in IEEE/ACM Transactions on Audio Speech and Language Processing, vol 26, no 8, pp 1434-1448, Jan 2018. Andreas I. Koutrouvelis had a significant contribution to Sections 8.3, 8.4, 8.5.1, 8.5.2 and 8.6.

In this chapter we present a practical application of a selection of the distributed optimization techniques presented in this thesis in the context of audio signal processing. We propose a new robust distributed linearly constrained beamformer which utilizes a set of linear equality constraints to reduce the cross power spectral density matrix to a block-diagonal form. The proposed beamformer has a convenient objective function for use in arbitrary distributed network topologies while having identical performance to a centralized implementation. Moreover, the new optimization problem is robust to relative acoustic transfer function (RATF) estimation errors and to target activity detection (TAD) errors. Two variants of the proposed beamformer are presented and evaluated in the context of multi-microphone speech enhancement in a wireless acoustic sensor network, and are compared with other state-of-the-art distributed beamformers in terms of communication costs and robustness to RATF estimation errors and TAD errors.

8.1. INTRODUCTION

Beamforming(see e.g., [126, 127, 128] for an overview) plays an important role in multi-microphone speech enhancement [129, 130, 131, 132]. The aim of a beamformer is the joint suppression of interfering noise and the preservation of an unknown target signal. The increasing usage of wireless portable devices equipped with microphones and limited power supplies, makes the notion of distributed beamforming in wireless acoustic sensor networks (WASNs) attractive compared to traditional centralized implementations [133]. The last decade, there are several proposed low-complexity distributed beamformers [134, 135, 136, 137, 138, 139, 140, 141, 142, 143] that mainly focus on achieving a good trade-off between noise reduction and communication cost.

Both centralized and distributed beamformers typically require an estimate of the cross-power spectral density matrix (CPSDM) of the noise/noisy measurements, and estimate(s) of the relative acoustic transfer function (RATF) vector(s) of the acoustic source(s) present in the acoustic scene. Estimation errors in these quantities result in performance degradation of beamformers. Much attention has therefore been given to the development of centralized robust beamformers which minimize the effects of RATF estimation errors (see e.g., [127, 128] for an overview). Developing robust *distributed* beamformers is more challenging than developing robust centralized beamformers, as distributed beamformers cannot afford high-complexity robust solutions. Therefore, it is desired to find very low-complexity robust distributed beamformers that achieve good performance trade-offs as described previously.

A low-complexity and easily manipulated family of beamformers are those that are calculated through linearly constrained quadratic problems such as: the minimum power distortionless responce (MPDR) beamformer [144] and its multiple constrained generalization, the linearly constrained minimum power (LCMP) beamformer [145]. Both beamformers minimize the total power of the *noisy* measurements while preserving the target. Therefore, their performance highly depends on the estimation accuracy of the RATF vector of the target source [127, 146, 128]. RATF estimation errors might result in removal of the actual target source and preservation in the direction of the wrongly estimated RATF vector.

Two straightforward, low-complexity, robust alternatives to MPDR and LCMP are the minimum variance distortionless response (MVDR) beamformer [146] and the linearly

constrained minimum variance (LCMV) beamformer [127], respectively. Both methods minimize the output *noise* power instead of the total noisy power, and thus require an estimate of the noise-only CPSDM. The noise CPSDM is typically estimated using a target activity detector (TAD) to identify target-free time-segments of audio. When the target is speech, this typically takes the form of a voice activity detector (see e.g., [131] for an overview). In [147], an alternative method was proposed to track the noise CPSDM also in time regions where the target is present. This method, however, highly depends on the estimation accuracy of the RATF vector of the target and its robustness to RATF estimation errors has not been tested.

Another family of low-complexity, robust alternatives to MPDR and LCMP are their diagonal loaded versions (see e.g., [148, 149, 150]). In both versions, the diagonal loading parameter, which is added to the main diagonal of the CPSDM, trades-off robustness against noise suppression. Specifically, by increasing the value of the diagonal loading parameter, a higher robustness to RATF estimation errors and a lower noise suppression is achieved. With diagonal loading, the use of a TAD is unnecessary. To the authors' knowledge, there are no low-complexity distributed approaches for choosing the optimal diagonal loading parameter. Additionally, a constant diagonal loading parameter will not be optimal for all acoustical scenarios and all frequency bins.

From the above it becomes clear that in addition to robustness and low-cost distributed calculations, LCMV and LCMP beamformers have the additional challenge of the RATF vector estimation of the target source and possibly the interferers. There are several centralized methods for RATF vector estimation (see e.g., [132] for an overview), however, there are yet no low-complexity distributed alternatives for arbitrary network topologies. In several applications, such as teleconferencing, the sources do not change their locations significantly over time and, therefore, one may estimate the RATF vectors of the target and/or the interferers only during initialization using a centralized approach and then use these estimated RATF vectors in the distributed beamformer. The slight positional errors that will most likely occur after this initial estimation require robust distributed beamformers. Note that in this paper, we mainly focus on this type of applications, i.e., the sources that do not significantly change their locations with respect to an initial reference location.

Notably, existing distributed beamformers can be classified based on how they address the issue of forming CPSDMs in WASNs. In the first class, the CPSDMs are approximated to form distributed implementations [134, 135, 136, 137] leading to approximately optimal performance. In the second class, the proposed beamformers obtain statistical optimality but do so at the expense of restricting the topology of the underlying WASN [138, 139, 140]. Statistically optimal beamformers which operate in unrestricted network topologies are much less common. An early example of such a beamformer is provided in [141], based on a maximum likelihood estimated LCMP beamformer. Unfortunately, this approach suffers from scaling communication costs as the number of samples used to construct the estimated CPSDM increases. In a similar vein, in [151], a distributed beamformer based on the pseudo-coherence principle was proposed. Similar to [141], the method in [151] can operate in cyclic networks. Furthermore, the authors demonstrated how the algorithm could perform near optimally with only a finite number of iterations, resulting in low transmission complexity. More recently, in [143] a topology

independent distributed beamformer (i.e. able to operate in cyclic networks) was proposed. Similar in its design to [139], this method requires very limited communication between nodes while guaranteeing convergence to the optimal beamformer. However, it was also demonstrated that the rate of this convergence was slow, requiring a large number of iterations to achieve this point. In practice, with even slowly varying sound fields such a rate of convergence may be detrimental to overall performance.

In this paper, we propose a new robust distributed linearly constrained beamformer, addressing the aforementioned challenges. The optimization problem of the proposed method nulls each interferer using a linear equality constraint, reducing the full-element noise or noisy CPSDM to a *block-diagonal* form. In contrast to MVDR, MPDR, LCMV and LCMP beamformers, the proposed objective function does not take into account correlation between different nodes in the WASN. Additionally, such an objective function is more convenient for distributed beamforming in WASNs of arbitrary topologies and significantly reduces the communication costs therein.

We show under realistic conditions, i.e., when the algorithms use non-ideally estimated RATF vectors and a non-ideal TAD, that the proposed method achieves a better predicted intelligibility than the MVDR and LCMV. The proposed method is less sensitive to RATF estimation errors, when TAD errors are not negligible, because of the block-diagonal form of the CPSDM.

The remainder of the paper is organized as follows. Section 8.2 presents the signal model. Section 8.3 reviews several methods of estimating the RATF vectors of the sources and the noisy/noise CPSDMs. Section 8.4 reviews the centralized and distributed linearly constrained beamformers. Section 8.5 presents the centralized and distributed versions of the proposed method. Section 8.6 shows the experimental results. Finally, concluding remarks are drawn in Section 8.7.

8.2. SIGNAL MODEL

Consider an arbitrary undirected WASN of N nodes. Without loss of generality, we assume that the underlying network (which is potentially cyclic) is connected. Denote by $V = \{1, \cdots, N\}$ the set of node indices and by E the set of edges of the network whereby $(i, j) \in E \iff i, j \in V$, $i \neq j$ can communicate with one another. Each node κ is equipped with M_κ microphones, where $\sum_{\kappa \in V} M_\kappa = M$, thus forming an M-element microphone array. One of the M microphones is selected as the reference microphone for the beamforming purpose. The distributed beamformers presented in this paper are formulated in the short-time Fourier transform (STFT) domain on a frame-by-frame basis. The noisy DFT coefficient of the j-th ($j = 1, \cdots, M$) microphone of the k-th frequency bin of the β-th frame is given by

$$y_j(k,\beta) = \underbrace{a_j(k,\beta)s(k,\beta)}_{x_j(k,\beta)} + \sum_{i=1}^{r} \underbrace{b_{ij}(k,\beta)v_i(k,\beta)}_{n_{ij}(k,\beta)} + u_j(k,\beta) \tag{8.1}$$

with $s(k,\beta)$ and $v_i(k,\beta)$ the target source and the i-th interferer at the reference microphone, $a_j(k,\beta)$ and $b_{ij}(k,\beta)$ the RATF vectors elements of each with respect to the j-th microphone, and $x_j(k,\beta)$, $n_{ij}(k,\beta)$ and $u_j(k,\beta)$ the target source, the i-th interferer and ambient noise at the j-th microphone. Note that the reference microphone

element of the RATF vectors is always equal to 1. Moreover, in the case of reverberant environments, the RATF vectors may also include a component due to early reverberation [152, 153]. Late reverberation and microphone self-noise are typically included in the ambient noise component. Note that even the late reverberation of the target has to be assigned to the ambient noise component because it reduces intelligibility [154, 155]. Thus, it should be reduced via the use of the beamformer. However, the early reflections (typically the first 50 ms [155]) are desired to be maintained because they typically contribute to intelligibility [154, 155]. Therefore, the ambient noise component is given by

$$u_j(k,\beta) = l_j^s(k,\beta) + \sum_{i=1}^{r} l_j^{v_i}(k,\beta) + c_j(k,\beta),$$

where $l_j^s(k,\beta)$ is the late reverberation component due to the target, $l_j^{v_i}(k,\beta)$ is the late reverberation component due to the i-th interferer, and $c_j(k,\beta)$ is the microphone self-noise.

In the sequel, we neglect the frame and frequency indices for the sake of brevity. Stacking all variables into vectors, Eq. (8.1) can be rewritten as

$$\mathbf{y} = \mathbf{x} + \underbrace{\sum_{i=1}^{r} \mathbf{n}_i + \mathbf{u}}_{\mathbf{n}} \in \mathbb{C}^{M\times 1}.$$

The CPSDM of $\mathbf{y}$ is given by $\mathbf{P}_\mathbf{y} = \mathrm{E}[\mathbf{y}\mathbf{y}^H]$, where $\mathrm{E}[\cdot]$ denotes statistical expectation. Assuming all sources are mutually uncorrelated, we have

$$\mathbf{P}_\mathbf{y} = \mathbf{P}_\mathbf{x} + \underbrace{\sum_{i=1}^{r} \mathbf{P}_{\mathbf{n}i} + \mathbf{P}_\mathbf{u}}_{\mathbf{P}_\mathbf{n}} \in \mathbb{C}^{M\times M}, \tag{8.2}$$

where $\mathbf{P}_\mathbf{x} = \mathrm{E}[\mathbf{x}\mathbf{x}^H] = p_s\mathbf{a}\mathbf{a}^H$ and $\mathbf{P}_{\mathbf{n}i} = \mathrm{E}[\mathbf{n}_i\mathbf{n}_i^H] = p_{v_i}\mathbf{b}_i\mathbf{b}_i^H$ are the CPSDMs of the target source and the i-th interferer at the microphones, respectively. Note that p_s and p_{v_i} are the power spectral densities of the target and the i-th interferer, respectively. Finally, the CPSDM of the ambient noise component, $\mathbf{P}_\mathbf{u}$, is given by

$$\mathbf{P}_\mathbf{u} = \mathrm{E}[\mathbf{u}\mathbf{u}^H] = \underbrace{\mathbf{P}_{\mathbf{l}s} + \sum_{i=1}^{r} \mathbf{P}_{\mathbf{l}u_i}}_{\mathbf{P}_\mathbf{l}} + \mathbf{P}_\mathbf{c} \in \mathbb{C}^{M\times M},$$

where $\mathbf{P}_\mathbf{l}$ denotes the CPSDM of the late reverberation, and $\mathbf{P}_\mathbf{c}$ the CPSDM of the microphone self-noise.

8.3. ESTIMATION OF SIGNAL MODEL PARAMETERS

The CPSDMs and the RATF vectors of the sources are unknown and have to be estimated in order to be available to the beamformers discussed in the sequel. In Sections 8.3.1 and 8.3.2, we review some existing methods for RATF vector and CPSDM estimation, respectively.

8.3.1. ESTIMATION OF RATF VECTORS

In practical applications, the true RATF vectors are reverberant due to room acoustics [153, 156, 157]. Several centralized methods have been proposed to estimate these RATF vectors (see e.g., [132] for an overview). In [153], the RATF vector of the target source is estimated by exploiting the assumption that the noise field is stationary. However, when the interferers are non-stationary, this can result in significant degradation in performance [156]. In [157] the subspaces of the target and interferers are estimated using a generalized eigenvalue decomposition (GEVD) combined with a TAD. While distributed methods have been proposed in the literature for performing GEVD-based subspace estimation in restricted network topologies (i.e., fully connected) [158], to our best knowledge, there are currently no distributed versions of the GEVD that operate in general cyclic networks.

In this work, we assume that estimates of the RATF vectors, $\hat{\mathbf{a}}$ and $\hat{\mathbf{b}}_i$, for $i = 1, \cdots, r$, are available at the initialization phase. In situations where the sources do not change their locations significantly with respect to an initial position, such as teleconferencing, the RATF vectors can be estimated (e.g., in a centralized way) during such an initialization. This will result in RATF estimation errors if the sources make some slight movements and, therefore, robust beamformers are required.

8.3.2. ESTIMATION OF CPSDMS

The LCMP and the MPDR beamformers depend on an estimate of the noisy CPSDM, $\hat{\mathbf{P}}_{\mathbf{y}}$. Typically, this estimate is computed using the sample average, which is given by

$$\hat{\mathbf{P}}_{\mathbf{y}} = \frac{1}{|L_y|} \sum_{l_y \in L_y} \mathbf{y}(l_y)\mathbf{y}^H(l_y),$$

where L_y is the set of frames of the entire time horizon and $|\cdot|$ denotes the cardinality of a set. The LCMV and the MVDR beamformers depend on an estimate of the noise CPSDM, $\hat{\mathbf{P}}_{\mathbf{n}}$. The noise CPSDM is estimated using the set of noise-only frames denoted by L_n, i.e.,

$$\hat{\mathbf{P}}_{\mathbf{n}} = \frac{1}{|L_n|} \sum_{l_n \in L_n} \mathbf{y}(l_n)\mathbf{y}^H(l_n),$$

where $|L_n| < |L_y|$. In order to obtain $\hat{\mathbf{P}}_{\mathbf{n}}$, a TAD is required to detect target presence/absence for each frame. The above two averages are updated in an online fashion, i.e., the average is updated for every frame using the average of the previous frame. This procedure becomes computationally demanding in a distributed context for two reasons. Firstly, the entire observation vector must be available at each time frame resulting in the need for data flooding. Secondly, that the storage of the entire CPSDM scales with the network size.

Estimation of the ambient noise CPSDM $\mathbf{P}_{\mathbf{u}}$ is a difficult task due to the late reverberation CPSDM $\mathbf{P}_{\mathbf{l}}$. Using a TAD it is nearly impossible to estimate $\mathbf{P}_{\mathbf{l}}$ alone. For sufficiently large rooms, the late reverberation is typically modelled as an ideal spherical isotropic noise field [159, 132].That is,

$$\hat{\mathbf{P}}_{\mathbf{l}} = \hat{p}_{\text{iso}}\mathbf{P}_{\text{iso}},$$

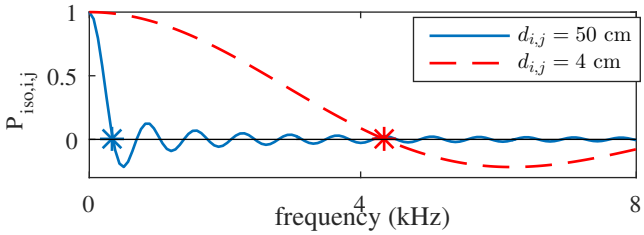


Figure 8.1: The spherically isotropic noise field correlation between two microphones i, j of distances $d_{i,j} = 4, 50$ cm and $f_s = 16$ kHz. The star marker denotes the first zero-crossing f_c.

where for the k-th frequency bin, the (i, j)-th element of $\mathbf{P}_{\text{iso}}$ is given by

$$\mathbf{P}_{\text{iso},i,j} = \text{sinc}\left(\frac{2\pi k f_s d_{i,j}}{\Phi c}\right),$$

where $d_{i,j}$ is the distance between microphones i and j, f_s is the sampling frequency, Φ is the number of frequency bins, and c is the speed of sound. The scaling $\hat{p}_{\text{iso}}$ can be estimated using several centralized methods (see e.g., [159]). To the best of our knowledge, there are no distributed methods for obtaining $\hat{p}_{\text{iso}}$.

Fig. 8.1 shows the values of the correlation function of Eq. (8.3.2) for various frequencies and distances $d_{i,j}$. The correlation can be roughly divided into two interesting frequency regions: one highly correlated on the left and one much less correlated on the right. The boundary between these regions occurs at the first zero-crossing given by $f_c = c/(2d_{i,j})$. It is clear that, the larger $d_{i,j}$ becomes, the smaller f_c is.

The CPSDM of the microphone self-noise, $\mathbf{P}_{\mathbf{c}} = c\mathbf{I}$ (where c is the power at each microphone), can be estimated in silent frames only (i.e., neither target nor interferers are active).

8.4. LINEARLY CONSTRAINED BEAMFORMING

Most linearly constrained beamformers are obtained from the following general optimization problem [145, 126, 127]

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \mathbf{w}^H \mathbf{P} \mathbf{w} \text{ s.t. } \mathbf{w}^H \Lambda = \mathbf{f}^H,$$

where $\Lambda \in \mathbb{C}^{M \times d}$, $\mathbf{f} \in \mathbb{C}^{d \times 1}$, and $\mathbf{P} \in \mathbb{C}^{M \times M}$ is typically the CPSDM of the noise or noisy measurements. The d constraints used in the optimization problem of Eq. (8.4) include at least the distortionless constraint for the target source, i.e., $\mathbf{w}^H \mathbf{a} = 1$, and, commonly, the nulling of the interferers, $\mathbf{w}^H \mathbf{b}_i = 0$ [126, 157, 160]. If we assume that $r < M - 1$, the linearly constrained beamformer can null all interferers and still have control on the minimization of the objective function. In this case, Λ and $\mathbf{f}$ are given by

$$\Lambda = \begin{bmatrix} \mathbf{a} & \mathbf{b}_1 & \cdots & \mathbf{b}_r \end{bmatrix}, \text{ and } \mathbf{f} = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^H. \tag{8.3}$$

It should be noted that by increasing the number of nulling constraints, the ambient output noise power may be boosted. The boost depends on the locations of the interferers [127] and the number of available degrees of freedom ($M-r-1$). However, in applications when $r \ll M-1$ this impact is much less significant. If $r < M-1$ and $\mathbf{P}$ is invertible, the optimization problem in Eq. (8.4), using the constraints in Eq. (8.3), has a closed-form solution given by [127]

$$\hat{\mathbf{w}} = \mathbf{P}^{-1}\Lambda\left(\Lambda^H\mathbf{P}^{-1}\Lambda\right)^{-1}\mathbf{f}.$$

When $\mathbf{P} = \mathbf{P_y}$, the linearly constrained beamformer takes the form of the LCMP beamformer given by

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \mathbf{w}^H\mathbf{P_y}\mathbf{w} \text{ s.t. } \mathbf{w}^H\Lambda = \mathbf{f}^H, \tag{8.4}$$

while if $\mathbf{P} = \mathbf{P_n}$, the LCMV is obtained and is given by

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \mathbf{w}^H\mathbf{P_n}\mathbf{w} \text{ s.t. } \mathbf{w}^H\Lambda = \mathbf{f}^H.$$

In the sequel, when we use the acronyms LCMV and LCMP we mean the LCMV and LCMP versions with the constraints given in Eq. (8.3). Another interesting linearly constrained beamformer is the one that has only the ambient noise component in the objective function [161], i.e.,

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \mathbf{w}^H\mathbf{P_u}\mathbf{w} \text{ s.t. } \mathbf{w}^H\Lambda = \mathbf{f}^H. \tag{8.5}$$

In this paper, we will refer to the linearly constrained beamformer in Eq. (8.5) as the ambient LCMV (ALCMV).

Using Eq. (8.2), the objective function of the LCMP problem, as noted in Eq. (8.4), is given by

$$\mathbf{w}^H\mathbf{P_y}\mathbf{w} = p_s\mathbf{w}^H\mathbf{a}\mathbf{a}^H\mathbf{w} + \sum_{i=1}^{r} p_{v_i}\mathbf{w}^H\mathbf{b}_i\mathbf{b}_i^H\mathbf{w} + \mathbf{w}^H\mathbf{P_u}\mathbf{w}.$$

Due to the included constraints in the LCMP (see Eq. (8.3)), the contributions of the early components of the sources to the objective function of Eq. (8.4) are constant. Thus, if $\hat{\mathbf{P}}_\mathbf{y} = \mathbf{P_y}$, $\hat{\mathbf{P}}_\mathbf{n} = \mathbf{P_n}$, $\hat{\mathbf{P}}_\mathbf{u} = \mathbf{P_u}$, and $\hat{\Lambda} = \Lambda$, the LCMP, LCMV and ALCMV beamformers are all equivalent. In practice, this never happens as there are always RATF estimation errors and CPSDM estimation errors, as explained previously.

8.4.1. RATF ESTIMATION ERRORS

There are two interesting cases. In the first case, if $\hat{\mathbf{P}}_\mathbf{y} = \mathbf{P_y}$, $\hat{\mathbf{P}}_\mathbf{n} = \mathbf{P_n}$, and $\hat{\mathbf{a}} = \mathbf{a}$, LCMP is equivalent to LCMV [127]. However, if $\hat{\mathbf{a}} \neq \mathbf{a}$, the LCMV beamformer (provided that $\hat{\mathbf{P}}_\mathbf{n}$ is accurately estimated), is more robust than the LCMP [127]. This is because LCMP will try to remove the actual target related to the RATF $\mathbf{a}$ as this is included in $\mathbf{P_y}$, while the preservation constraint is on the wrongly estimated $\hat{\mathbf{a}}$. However, if there are also TAD errors, $\hat{\mathbf{P}}_\mathbf{n}$ may also contain portions of $\mathbf{P_x}$ and, as a result, the LCMV may also have severe performance degradation like the LCMP.

In the second case, if $\hat{\mathbf{P}}_\mathbf{n} = \mathbf{P_n}$, $\hat{\mathbf{P}}_\mathbf{u} = \mathbf{P_u}$, and $\hat{\mathbf{b}}_i = \mathbf{b}_i$, for $i = 1, \cdots, r$, LCMV is equivalent to ALCMV. However, if any of the $\hat{\mathbf{b}}_i$'s contain estimation errors, there will be power

leakage of the corresponding interferer(s), which is not controllable, neither by the objective function nor by the constraints of the ALCMV problem in Eq. (8.5). Moreover, if there are interferers whose RATF vectors have not been placed in the constraints, the ALCMV will also be unable to reduce them in a controlled way. In contrast, if $\hat{\mathbf{P}}_{\mathbf{n}}$ is estimated accurately, the LCMV will reduce these power leakages. In this case, the LCMV will most likely have a better noise reduction performance than its ALCMV counterpart.

We can conclude that the performance degradation of linearly constrained beamformers due to RATF estimation errors is mainly influenced by the selection of the CPSDM, $\mathbf{P}$, in the objective function of Eq. (8.4). A low-cost robust linearly constrained beamformer should have good performance under both RATF estimation errors and TAD errors. There are several approaches to achieve this. The most popular is via diagonal loading of $\mathbf{P}$. However, to the authors' knowledge there are no low-cost distributed approaches for optimally selecting the diagonal loading value. Another robust low-cost option is to use a fixed superdirective linearly constrained beamformer, i.e., a linearly constrained beamformer with a (semi)fixed $\mathbf{P}$ [130]. A fixed linearly constrained beamformer does not use a TAD and guarantees that there will not be any portion of $\mathbf{P}_{\mathbf{x}}$ in $\mathbf{P}$. Two interesting fixed linearly constrained beamformers are discussed in the next section.

8.4.2. FIXED SUPERDIRECTIVE LINEARLY CONSTRAINED BEAMFORMERS

The fixed superdirective beamformers [130] assume a certain noise field and use in the objective function a certain coherence function like the one in Eq. (8.3.2). Since the early components of the interferers can be nullified using a linearly constrained beamformer, the noise field that remains is the late reverberation as explained previously in this section. Recall from Section 8.3.2, that the estimation of $\mathbf{P}_{\mathbf{u}}$ is a difficult task due to the CPSDM of the late reverberation, $\mathbf{P}_{\mathbf{l}}$. Typically, in the literature (see e.g., [162, 130, 163]) models of $\mathbf{P}_{\mathbf{l}}$ are used in beamformers instead. The most common choice is to use $\mathbf{P}_{\text{iso}}$. If one chooses $\mathbf{P} = \mathbf{P}_{\text{iso}}$, the microphone self-noise will be boosted in low frequencies [130]. Thus, a diagonal-loaded version is typically used [164, 130], i.e.,

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \mathbf{w}^H (p_{\text{iso}} \mathbf{P}_{\text{iso}} + \mathbf{P}_{\mathbf{c}}) \mathbf{w} \text{ s.t. } \mathbf{w}^H \Lambda = \mathbf{f}^H,$$

where $\mathbf{P}_{\mathbf{c}} = c\mathbf{I}$ (see Section 8.3.2). Although, the microphone-self noise power, c, typically remains constant over time, p_{iso} changes. To the best of our knowledge, there are no distributed estimation methods of the scaling coefficient p_{iso}. We call the beamformer in Eq. (8.4.2) as isotropic LCMV (ILCMV).

Another popular fixed linearly constrained beamformer uses in the objective function the most simplistic option which is $\mathbf{P} = \mathbf{I}$, i.e.,

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \mathbf{w}^H \mathbf{w} \text{ s.t. } \mathbf{w}^H \Lambda = \mathbf{f}^H.$$

In this paper, we will refer to this as the linearly constrained delay and sum (LCDS) beamformer. It is identical to the fixed beamformer of the generalized side-lobe canceller implementation of the LCMP beamformer (using the constraints in Eq. (8.3)) in [157]. Unlike ILCMV, the LCDS is easily distributable due to the separable nature of the objective function. This can be achieved via similar methods to those demonstrated in Section

8.5.3 and need only be performed once. Following this, the output can be computed via data aggregation or by solving a simple averaging problem, again lending itself to distributed implementations.

Similar to ALCMV, the ILCMV and LCDS beamformers cannot control power leakages due to inaccurate estimates of the interferers' RATF vectors and cannot control interferers which are not included in the constraints.

8.4.3. OTHER RELATED LINEARLY CONSTRAINED BEAMFORMERS

If we skip the nulling constraints and only impose the target distortionless constraint, the LCMV (LCMP) reduces to the MVDR (MPDR) [144, 126]. Similar to LCMV and LCMP, MVDR and MPDR are equivalent under the assumption that $\hat{\mathbf{P}}_{\mathbf{y}} = \mathbf{P}_{\mathbf{y}}$ and $\hat{\mathbf{P}}_{\mathbf{n}} = \mathbf{P}_{\mathbf{n}}$ and $\hat{\mathbf{a}} = \mathbf{a}$ [127]. However, when $\hat{\mathbf{a}} \neq \mathbf{a}$, the MVDR is more robust to RATF estimation errors [146, 127]. A special case of the MPDR is the delay and sum (DS) beamformer [152] which replaces the noisy CPSDM with the identity matrix. The DS has worse performance compared to the MVDR (MPDR) in correlated noise fields but results in very robust performance to RATF estimation errors [146] and TAD errors.

8.4.4. DISTRIBUTED LINEARLY CONSTRAINED BEAMFORMERS

The development of distributed beamformers has focused on adapting LCMV (LCMP) based approaches for use in WASNs. However, this adaptation has not come without additional challenges [165]. Most notable is the limited communication between devices which makes the formation of estimated CPSDMs nearly impossible without the use of a fusion center [133]. To address this, two main classes of distributed beamformers have appeared in the literature: approximately optimal variants and optimal approaches which operate in certain networks.

One such sub-optimal variant is the distributed DS beamformer introduced in [134]. Based on randomised gossip [12], this low-cost method operates in general cyclic networks but fails to exploit spatial correlation to improve noise reduction. In contrast, distributed approximations of the MVDR beamformer [135, 136] assume that disjoint nodes are uncorrelated essentially masking the true CPSDMs. While lending themselves to distributed implementations, such approaches fail to take into account the true correlations between observed signals across the network, resulting in sub-optimal performance.

By restricting the network topology, typically to be acyclic or fully connected, optimal distributed beamformers have been proposed. These algorithms [139, 140] exploit efficient data aggregation to construct global beamformers from a composition of local filters and have been shown to be iteratively optimal. However, the additional communication overhead required to maintain a constant network topology across frames can be prohibitively expensive due to unpredictable network dynamics. Furthermore, such maintenance may be impossible in the case of node failure.

It is worth mentioning that it is not the use of an acyclic network in [139, 140] itself which is limiting, but rather the need for this network to be invariant over time. In [143], this point was exploited to form a fully distributed beamformer for use in general cyclic topologies. Like [139] and [140], [143] constructs a global beamformer as a composition of local beamformers at each node. Importantly, the method by which these local beam-

formers are combined does not depend on the underlying network topology. This allows the network to vary between frames, overcoming the need for maintaining a fixed topology in all time instances. The method in [143] was shown to be iteratively optimal with its main drawback being a decrease in convergence rate compared to [139], requiring a larger number of frames to obtain near optimal performance.

In contrast, in [141], an optimal distributed beamformer was proposed for use in cyclic networks by exploiting the structure of estimated CPSDMs to cast LCMP beamforming as distributed consensus. However, for CPSDM estimates based on a large number of frames, the proposed algorithm's communication cost scaled poorly. In contrast to [138, 139, 140] and [143], a benefit of [141] was that the proposed implementation was frame-optimal, i.e. that it obtained the performance of an equivalent centralized implementation in each frame. The beamformer proposed in [151] exploited a similar method of distributed implementation, but exploited the pseudo coherence principle of human speech to overcome the scaling communication costs found in [141].

The approaches of both [141] and [151] made use of internal optimization schemes which require a large number of iterations per frame to obtain optimal performance. However, in [151] it was shown that near optimal performance could be obtained using only a finite number of iterations of this internal solver. Such a result raises the question whether a similar approach could be employed as a general way of reducing the transmission costs associated with cyclic beamforming methods. For the beamformers proposed in this work, this point is touched upon in Section 8.5.7.

In contrast to the methods above, the beamformers proposed in Section 8.5 are fully distributable without imposing restrictions on the underlying network topology or scaling communication costs while also being optimally computable in each frame. In this way, the proposed methods combine the strengths of existing distributed beamformers while also avoiding their various limitations.

8.5. PROPOSED METHOD

In the previous section, we have highlighted the susceptibility of several existing beamformers to RATF estimation errors and TAD errors and the challenge of deploying these algorithms in distributed contexts. Here, we propose two different linearly constrained beamformers which are efficiently distributable for arbitrary network topologies, robust to RATF estimation errors and TAD errors, while at the same time are able to control the power leakage of the interferers.

Typically, the microphones within a node are nearby, while the microphones from different nodes are further away. Therefore, the late reverberation will be highly correlated in the first case, while in the latter less correlated (see Fig. 8.1). Therefore, providing that the nodes are sufficiently far away from each other, one may approximate the full element matrix $\mathbf{P_u}$ with the *block-diagonal* matrix $\bar{\mathbf{P}}_\mathbf{u}$ where every block corresponds to the CPSDM of the late reverberation of one node only and the microphone-self noise. Therefore, we propose the block-diagonal ALCMV (BDALCMV) which is given by

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \mathbf{w}^H \bar{\mathbf{P}}_\mathbf{u} \mathbf{w} \text{ s.t. } \mathbf{w}^H \Lambda = \mathbf{f}^H. \tag{8.6}$$

Note that if every node has only one microphone, $\bar{\mathbf{P}}_\mathbf{u}$ becomes diagonal. This block-

diagonalization lends itself to distributed implementations, reflecting a similar objective structure to that of the DS and LCDS beamformer.

While the proposed BDALCMV beamformer has a number of benefits from the perspective of distributed signal processing, like ALCMV, the challenge becomes the estimation of $\bar{\mathbf{P}}_{\mathbf{u}}$, and handling the possible power leakages of the interferers as in the case of DS, LCDS, ALCMV. Therefore, in Sections 8.5.1, and 8.5.2 we introduce two variations of the BDALCMV beamformer which do not require the estimation of $\bar{\mathbf{P}}_{\mathbf{u}}$ and are robust to power leakages of the interferers. Moreover, in Sections 8.5.3—8.5.7, we introduce distributed implementations of the proposed beamformers.

8.5.1. BDLCMP BEAMFORMER

The first proposed practical variant of BDALCMV is the BDLCMP which uses in the objective function the block-diagonal noisy CPSDM, $\bar{\mathbf{P}}_{\mathbf{y}}$. That is,

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \mathbf{w}^H \bar{\mathbf{P}}_{\mathbf{y}} \mathbf{w} \text{ s.t. } \mathbf{w}^H \Lambda = \mathbf{f}^H. \tag{8.7}$$

This results in a local estimation problem, which can be carried out independently at each node without the need of a TAD. This method handles the possible power leakages due to inaccurate estimates of the interferers' RATF vectors and can suppress the interferers that are not included in the constraints.

In case of RATF estimation errors of the target source, the BDLCMP will have similar problems to the LCMP because in the block-diagonal matrices, there will be portions of the corresponding target block-diagonal CPSDMs. However, the performance degradation will not be that great as with the LCMP. This can be easily explained by considering the extreme scenario of a fully correlated noise field in which we assume that $M > r+1$, $\hat{\mathbf{P}}_{\mathbf{y}} = \mathbf{P}_{\mathbf{y}}$, $\mathbf{P}_{\mathbf{u}} \approx 0$, $\hat{\mathbf{b}}_i = \mathbf{b}_i, i = 1, \cdots, r$ and $\hat{\mathbf{a}} \neq \mathbf{a}$. In this case, the optimization problem of LCMP in Eq. (8.4) will be approximately equivalent[1] to the following optimization problem:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \mathbf{w}^H \hat{\mathbf{P}}_{\mathbf{y}} \mathbf{w} \text{ s.t. } \mathbf{w}^H \tilde{\Lambda} = \tilde{\mathbf{f}}^H,$$

where

$$\tilde{\Lambda} = \left[\hat{\mathbf{a}} \ \mathbf{a} \ \hat{\mathbf{b}}_1 \ \cdots \ \hat{\mathbf{b}}_r\right], \text{ and } \tilde{\mathbf{f}}^H = \left[1 \ 0 \ 0 \cdots 0\right].$$

That is, the LCMP will approximately nullify the target source. In contrast, due to the block-diagonal CPSDM, the BDLCMP will approximately nullify the target source iff $M > rN + 2r + 1$, where N is the number of nodes. Specifically, if $M > rN + 2r + 1$ is satisfied, the BDLCMP will be approximately equivalent to the following optimization problem:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \mathbf{w}^H \hat{\bar{\mathbf{P}}}_{\mathbf{y}} \mathbf{w} \text{ s.t. } \mathbf{w}^H \tilde{\Lambda} = \tilde{\mathbf{f}}^H,$$

where

$$\begin{aligned}
\tilde{\Lambda} &= \left[\hat{\mathbf{a}} \ \tilde{\mathbf{a}}_1 \ \tilde{\mathbf{a}}_2 \ \cdots \ \tilde{\mathbf{a}}_N \ \hat{\mathbf{b}}_1 \cdots \hat{\mathbf{b}}_r \ \tilde{\mathbf{b}}_{11} \cdots \tilde{\mathbf{b}}_{1N} \cdots \tilde{\mathbf{b}}_{r1} \cdots \tilde{\mathbf{b}}_{rN}\right], \\
\tilde{\mathbf{f}}^H &= \left[1 \ 0 \ 0 \cdots 0\right] \\
\tilde{\mathbf{a}}_i &= \left[\mathbf{0} \ \mathbf{a}_i \ \mathbf{0}\right]^H, \quad \tilde{\mathbf{b}}_{ji} = \left[\mathbf{0} \ \mathbf{b}_{ji} \ \mathbf{0}\right]^H \in \mathbb{C}^{M \times 1}.
\end{aligned}$$

[1] It is approximately equivalent because $\mathbf{P}_{\mathbf{u}} \approx 0$. Moreover, the target RATF estimation errors should be sufficiently large.

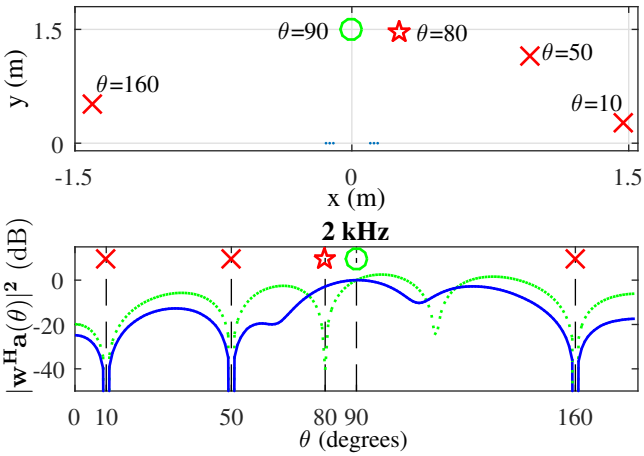


Figure 8.2: Example: three interferers (with marker 'x') and one target (with marker $\star$) at 80°. The RATF vector of the target points at 90°. The directivity pattern, $|\mathbf{w}^H\mathbf{a}(\theta)|^2$ (in dB), is computed in the range $0° \leq \theta \leq 180°$, for BDLCMP (solid line) and LCMP (dotted line), for the frequency 2 kHz.

Here $\mathbf{a}_i, \mathbf{b}_{ji}$ are the elements of the RATF vector $\mathbf{a}, \mathbf{b}_j$ corresponding to node i, respectively. Note that for $M < rN + 2r + 1$ the BDLCMP will not have enough degrees of freedom to achieve $\mathbf{w}^H\tilde{\mathbf{a}}_i = 0$ $(i = 1, \cdots, N)$ and, thus, will not nullify the target signal. Thus, more microphones are needed in the BDLCMP beamformer to nullify the target signal compared to the LCMP beamformer. Hence, the BDLCMP is more robust to target RATF estimation errors compared to the LCMP for the same number of microphones M, when $M < rN + 2r + 1$, in this particular scenario of a fully correlated noise field. In more general noise fields, where $\mathbf{P_u}$ is not negligible, both LCMP and BDLCMP will not nullify the target using the same finite number of microphones. However, LCMP will suppress more the target signal than the BDLCMP, because the first exploits the full-element noisy CPSDM matrix.

Fig. 8.2 shows the directivity patterns of LCMP and BDLCMP for a simple acoustic scenario with a linear microphone array separated into two nodes where each node has three microphones. The target source is at 80°, but the estimated RATF vector of the target is at 90°. The interferers and their RATF vectors are at 10°, 50° and 160°. All RATF vectors are anechoic in this example and there is a slight amount of microphone-self noise. It is clear from the directivity pattern in Fig. 8.2, that LCMP suppresses the target signal significantly, while BDLCMP does not.

It is worth mentioning that if $\hat{\mathbf{b}}_i \neq \mathbf{b}_i$, it easy to show (following the same steps as before) that the LCMP will typically suppress more the i-th interferer than BDLCMP, if both use the same number of microphones. This means that the power leakages of the interferers will be suppressed more with the LCMP compared to the BDLCMP. Nevertheless, we will experimentally show in Section 8.6, that the final intelligibility improvement of BDLCMP is much greater than the LCMP, because BDLCMP distorts much less the target.

8.5.2. BDLCMV BEAMFORMER

To further increase the robustness of the proposed method, we introduce the BDLCMV variant which uses in the objective function the block-diagonal version of the noise CPSDM, $\bar{\mathbf{P}}_{\mathbf{n}}$. Therefore, the BDLCMV is given by

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \mathbf{w}^H \bar{\mathbf{P}}_{\mathbf{n}} \mathbf{w} \text{ s.t. } \mathbf{w}^H \Lambda = \mathbf{f}^H. \tag{8.8}$$

Similar to the relationship between LCMV and LCMP, the BDLCMV typically enjoys more robustness than the BDLCMP when $\bar{\mathbf{P}}_{\mathbf{n}}$ is estimated accurately enough. However, when there are TAD errors, we will show that the performance gap reduces between the two methods. The BDLCMV also handles the possible power leakages of the interferers, and can suppress the interferers that are not included in the constraints.

If each node has only one microphone, then BDLCMV becomes diagonal. In this case, it can be viewed as a weighted version of the LCDS beamformer, and without nulling constraints, can be viewed as a weighted DS beamformer.

8.5.3. DISTRIBUTED IMPLEMENTATION OF THE PROPOSED METHOD

Given a block-diagonal matrix $\bar{\mathbf{P}}$, which can be $\bar{\mathbf{P}}_{\mathbf{u}}$, $\bar{\mathbf{P}}_{\mathbf{n}}$ or $\bar{\mathbf{P}}_{\mathbf{y}}$, and a known constraint matrix Λ, we now demonstrate how we can form a distributed version of the proposed methods for use in general cyclic networks by using a similar technique to that presented in [141]. Importantly, the imposed block diagonal structure of the estimated CPSDM results in a naturally separable objective function, leading to a substantial reduction in communication costs compared to those in [141]. To demonstrate this, denote by $\mathbf{w}_k$, Λ_k and $\bar{\mathbf{P}}_k$ the elements of $\mathbf{w}$, the rows of Λ and the block diagonal component of $\bar{\mathbf{P}}$ associated with node k, respectively. Eqs. (8.6), (8.7) and (8.8) can therefore be rewritten as

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \frac{1}{2} \sum_{k=1}^{N} \mathbf{w}_k^H \bar{\mathbf{P}}_k \mathbf{w}_k \text{ s.t. } \sum_{k=1}^{N} \mathbf{w}_k^H \Lambda_k = \mathbf{f}^H.$$

8

The real-valued Lagrangian of this problem is given by

$$\mathscr{L}(\mathbf{w}, \boldsymbol{\mu}) = \sum_{k=1}^{N} \left(\frac{\mathbf{w}_k^H \bar{\mathbf{P}}_k \mathbf{w}_k}{2} - \Re\left(\boldsymbol{\mu}^H \left(\Lambda_k^H \mathbf{w}_k - \frac{\mathbf{f}}{N} \right) \right) \right),$$

where we have partitioned the constraint vector $\mathbf{f}$ into N equal parts, $\mathbf{f}/N$,one for each node $i \in V$. Taking complex partial derivatives [166], it follows that

$$\hat{\mathbf{w}_k} = \bar{\mathbf{P}}_k^{-1} \Lambda_k \boldsymbol{\mu}, \tag{8.9}$$

such that the corresponding dual function is thus given by

$$q(\boldsymbol{\mu}) = -\sum_{k=1}^{N} \frac{\boldsymbol{\mu}^H \Lambda_k^H \bar{\mathbf{P}}_k^{-1} \Lambda_k \boldsymbol{\mu}}{2} + \Re\left(\boldsymbol{\mu}^H \mathbf{f} \right).$$

The resulting dual optimization problem is given by

$$\hat{\boldsymbol{\mu}} = \arg\min_{\boldsymbol{\mu}} \sum_{k=1}^{N} \left(\frac{\boldsymbol{\mu}^H \Lambda_k^H \bar{\mathbf{P}}_k^{-1} \Lambda_k \boldsymbol{\mu}}{2} - \Re\left(\boldsymbol{\mu}^H \frac{\mathbf{f}}{N} \right) \right). \tag{8.10}$$

8.5.4. ACYCLIC IMPLEMENTATION VIA MESSAGE PASSING

We begin by demonstrating how, when the underlying network is acyclic (tree structured), the problem in Eq. (8.10) can be solved in a distributed manner. Similar to the approach introduced in [143], there is no need for this acyclic network to be constant between frames, allowing it to adapt to the time-varying connectivity of dynamic networks. This contrasts [139, 140] where the network topology must remain constant.

In the following, we consider two different approaches to compute the optimal $\boldsymbol{\mu}$ in tree structured networks. In the first approach, we exploit the fact that Eq. (8.10) can be directly solved by aggregating the sum of the local matrices $\frac{1}{2}\Lambda_\kappa^H \bar{\mathbf{P}}_\kappa^{-1}\Lambda_\kappa$ to a common location. In the case of acyclic networks, this aggregation can be performed efficiently with the common location forming the root node of the network. This root node can simply be a point in the network where we choose to extract the beamformer output signal.

To sketch the process of this data aggregation, we partition the set of neighbors of each node κ into two groups. The first group, denoted by $\mathscr{C}_\kappa$, represents the set of children of node κ. The second set, which is a unique node identifier, is the parent of node κ denoted by $\mathscr{P}_\kappa$. In particular, $\mathscr{P}_\kappa \cup \mathscr{C}_\kappa = \mathscr{N}(\kappa)\ \forall \kappa \in V$, where $\mathscr{N}(\kappa) = \{\iota \mid (\kappa,\iota) \in E\}$. Note that for the root node $\mathscr{P}_\kappa = \emptyset$. These sets can be determined per frame by selecting a root node and forming a spanning tree via a breadth-first or depth-first search.

Once these sets are known, the process begins at the leaf nodes of the networks (those nodes for which $\mathscr{C}_\kappa = \emptyset$) and consists of the transmission of a message from these nodes (κ) to their parents ($\mathscr{P}_\kappa$). The aggregation messages are matrices and take the form

$$\mathbf{M}_{\kappa\to\mathscr{P}_\kappa} = \frac{\Lambda_\kappa^H \bar{\mathbf{P}}_\kappa^{-1}\Lambda_\kappa}{2}.$$

Of the set of remaining nodes, those nodes which have received a message from all but one of their neighbors can repeat this process (the remaining neighbor is their parent node). Their messages take a more general form given by

$$\mathbf{M}_{i\to\mathscr{P}_i} = \frac{\Lambda_i^H \bar{\mathbf{P}}_i^{-1}\Lambda_i}{2} + \sum_{k\in\mathscr{C}_i} \mathbf{M}_{k\to i},$$

whereby local information at each node is first combined with that from their children. This process is repeated until the root node has received messages from all its children at which point the aggregation operation is complete.

Due to their positive semidefinite structure, the transmission of each message per node comprises $\frac{1}{2}((r+1)^2+r+1)$ unique variables resulting in a total of $\frac{1}{2}(r^2+3r+2)(N-1)$ transmitted variables for each frequency bin per frame. The optimal dual variables can then be diffused back into the network to allow the optimal beamformer weight vector to be computed at each node in parallel. This additional diffusion stage results in a further $(r+1)(N-K)$ transmitted variables where K denotes the number of leaf nodes. The beamformer output can then be computed by simply aggregating the sum $\sum_{i\in V}\mathbf{w}_i^H\mathbf{y}_i$ through the network, incurring a total cost of $(N-1)$ transmissions per frequency bin. Finally, if the estimate of $\bar{\mathbf{P}}$ does not change between frames, i.e., $\Delta\bar{\mathbf{P}} = \mathbf{0}$, the estimated weight vector need not be recomputed. An example of this occurs in noisy frames for the

proposed BDLCMV method, reducing the cost of this algorithm in such frames to that of simply computing the beamformer output.

8.5.5. CYCLIC WEIGHT VECTOR COMPUTATION VIA PDMM

For more general network structures, Eq. (8.10) can be transformed to a fully distributable form. To do so, we introduce local versions of $\boldsymbol{\mu}$ at each node, denoted by $\boldsymbol{\mu}_\kappa$, and impose that $\boldsymbol{\mu}_\kappa = \boldsymbol{\mu}_\iota \; \forall \; (\kappa,\iota) \in E$. The resulting problem is given by

$$\hat{\boldsymbol{\mu}} = \arg\min_{\boldsymbol{\mu}} \sum_{\kappa=1}^{N} \left(\frac{\boldsymbol{\mu}_\kappa^H \Lambda_\kappa^H \bar{\mathbf{P}}_\kappa^{-1} \Lambda_\kappa \boldsymbol{\mu}_\kappa}{2} - \Re\left(\boldsymbol{\mu}_\kappa^H \frac{\mathbf{f}}{N}\right) \right) \quad \text{s.t. } \boldsymbol{\mu}_\kappa = \boldsymbol{\mu}_\iota \quad \forall (\kappa,\iota) \in E. \tag{8.11}$$

Note that at optimality, this problem is entirely equivalent to the problem in Eq. (8.10), assuming the network is connected. Due to its separable quadratic structure, Eq. (8.11) can be solved via a wide range of existing distributed solvers [**?**, **?**, 96]. In this work, we consider solving Eq. (8.11) using the primal dual method of multipliers (PDMM) proposed in [96].

To define the PDMM updating scheme, we begin by again considering the equivalent graph representation of the network, parameterised by node set V and edge set E. For each node κ and edge $(\kappa,\iota) \in E$, define the vectors $\boldsymbol{\mu}_\kappa^{(0)} = \boldsymbol{\gamma}_{\kappa,\iota}^{(0)} = \mathbf{0} \in \mathbb{C}^{\mathbf{r}+1}$, $\forall \kappa = 1,\ldots,N$, $(\kappa,\iota) \in E$ respectively. As per the PDMM algorithm in [96], the optimizers of Eq. (8.11) can then be computed by iteratively updating the dual variables ($\boldsymbol{\mu}_\kappa$) and directed edge variables ($\boldsymbol{\gamma}_{\kappa|\iota}$) as

$$\begin{aligned}
\boldsymbol{\mu}_\kappa^{(t+1)} &= \left(\Lambda_\kappa^H \bar{\mathbf{P}}_\kappa^{-1} \Lambda_\kappa + \rho |\mathscr{N}(\kappa)| \mathbf{I} \right)^{-1} \left(\frac{\mathbf{f}}{N} + \sum_{\iota \in \mathscr{N}(\kappa)} \left(\frac{\kappa - \iota}{|\kappa - \iota|} \boldsymbol{\gamma}_{\iota|\kappa}^{(t)} + \rho \boldsymbol{\mu}_\iota^{(t)} \right) \right) \\
\boldsymbol{\gamma}_{\kappa|\iota}^{(t+1)} &= \boldsymbol{\gamma}_{\iota|\kappa}^{(t)} - \rho \frac{\kappa - \iota}{|\kappa - \iota|} \left(\boldsymbol{\mu}_\kappa^{(t+1)} - \boldsymbol{\mu}_\iota^{(t)} \right),
\end{aligned}$$

where each $\rho \in (0, +\infty)$ is the step size for the iterative algorithm and t denotes the iteration index. The notation $\kappa|\iota$ is used to define the edge variable computed at node κ related to the edge $(\kappa,\iota) \in E$.

The edge based update requires the transmission of information between neighbouring nodes, as can be noted in the dependence of $\boldsymbol{\gamma}_{\kappa|\iota}^{(t+1)}$ on $\boldsymbol{\gamma}_{\iota|\kappa}^{(t)}$ and $\boldsymbol{\mu}_\iota^{(t)}$. As highlighted in [96] however, this only requires the transmission of the $\boldsymbol{\mu}_\kappa$ variables and, thus, can be performed via a broadcast transmission protocol at each node. These updates can then be iterated until a desired level of precision is achieved after which $\hat{\mathbf{w}}_j$ can be calculated locally at each node via Eq. (8.9).

Each iteration of the proposed algorithm requires the transmission of $r+1$ variables per node. In an existing optimal cyclic beamformer [141] this cost was $r+1+|L_y|$, where $|L_y|$ is the number of frames used to form a maximum likelihood estimated version of the CPSDM. The proposed method therefore requires $|L_y|$ less transmissions per iteration, resulting in a substantial saving in transmission costs.

8.5.6. BEAMFORMER OUTPUT COMPUTATION

Once the weight vector is known, the beamformer output can then be computed via various distributed averaging techniques (see [38] for an overview). In the case of this

work we again consider the use of PDMM for this task. Consider the standard distributed averaging problem given by

$$\min_{\mathbf{x}} \frac{1}{2}\sum_{\kappa=1}^{N} \|\mathbf{x}_\kappa - \mathbf{w}_\kappa^H \mathbf{y}_\kappa\|^2 \quad \text{s.t. } \mathbf{x}_\kappa = \mathbf{x}_\iota \; \forall (\kappa,\iota) \in E. \tag{8.12}$$

Again, from [96], the PDMM update equations for this problem are given by

$$\begin{aligned}
\mathbf{x}_\kappa^{(t+1)} &= \frac{\left(\mathbf{w}_\kappa^H \mathbf{y}_\kappa + \sum_{\iota\in\mathcal{N}(\kappa)} \left(\frac{\kappa-\iota}{|\kappa-\iota|}\mathbf{z}_{\iota|\kappa}^{(t)} + \rho \mathbf{x}_\iota^{(t)}\right)\right)}{1+\rho|\mathcal{N}(\kappa)|} \\
\mathbf{z}_{\kappa|\iota}^{(t+1)} &= \mathbf{z}_{\iota|\kappa}^{(t)} - \rho\frac{\kappa-\iota}{|\kappa-\iota|}\left(\mathbf{x}_\kappa^{(t+1)} - \mathbf{x}_\iota^{(t)}\right),
\end{aligned}$$

where $\mathbf{z}_{\kappa|\iota}$ denotes the directed edge variable owned by node κ. By iterating these updates, every node in the network can learn the average of the vector $\mathbf{w}^H\mathbf{y}$. Once the average is known, this can be scaled by a factor of N to recover the beamformer output. Alternatively, we can employ the same acyclic beamformer output computation approach as used in Sec. 8.5.4. While this removes the entirely cyclic nature of the algorithm as the tree structured network used can change in each frame, the overhead of using an acyclic network is still substantially reduced in contrast to the work of [139, 140].

8.5.7. CYCLIC BEAMFORMING WITH FINITE NUMBERS OF ITERATIONS

In general distributed applications, deterministic signal processing is desirable. This point is even more pressing in the case of distributed audio processing. Thus, an unbounded requirement on the iteration count of an algorithm is cumbersome. Unfortunately, in practice, the total number of transmissions required to solve the problems in Eq. (8.11) and (8.12), via general cyclic solvers such as PDMM, is dependent not only on the choice of the solver but also on the WASN topology. As such, it is not possible to analytically bound this transmission cost for arbitrary networks. However, in the distributed beamforming method presented in [151], which also used PDMM as a solver, it was found that near optimal performance was achieved in only a limited number iterations. In this way it is expected that the number of iterations required to achieve a good level of performance is not unnecessarily large. As such we can impose a hard limit on the number of iterations performed without significantly degrading performance.

An additional observation is that, due to its dependence on a recursively averaged covariance matrix, the weight vector $\mathbf{w}$ will vary smoothly with time. With regards to the PDMM algorithm, this corresponds to the fact that both the dual and edge variables will also vary somewhat smoothly. As such, one way to improve precision even under the scenario of a finite number of iterations it to use a warm-start procedure. Defining the maximum number of iterations by $t_{\max}$, this warm-start procedure is implemented by setting

$$\boldsymbol{\mu}_\beta^{(0)} = \boldsymbol{\mu}_{\beta-1}^{(t_{\max})} \quad \text{and} \quad \boldsymbol{\gamma}_{\beta,\kappa|\iota}^{(0)} = \boldsymbol{\gamma}_{\beta-1,\kappa|\iota}^{(t_{\max})}, \tag{8.13}$$

where the additional subscript denotes the frame index β. In the case of a constant CPSDM estimate this procedure allows the finite iterations in multiple frames to be used to solve the same problem i.e. a higher precision weight vector can be achieved. In

Table 8.1: Transmission costs of distributed beamformers in dynamic sound fields. N denotes the number of nodes, K denotes the number of leaf nodes, r denotes the number of interferers, and $t_{\max}$ denotes the maximum number of iterations.

Beamformer Weight Vector Computation	
Algorithm	Transmissions per frame & frequency bin
BDLCMV/BDLCMP (Cyclic)	$t_{\max}(r+1)N$
BDLCMV/BDLCMP (Acyclic)	$\frac{1}{2}(r^2+3r+2)(N-1)+(r+1)(N-K)$
BDLCMV (Acyclic $\Delta\bar{\mathbf{P}}=\mathbf{0}$)	0
DLCMV (Acyclic) [139]	$(2N-1-K)$
DGSC (Acyclic) [140]	$(2N-1-K)+(r+1)(N-K)$
TI-DANSE (Cyclic) [143]	$(2N-1-K)(r+1)$
Beamformer Output Computation	
Algorithm	Transmissions per frame & frequency bin
Cyclic	$t_{\max}N$
Acyclic	$N-1$

the case of slowly varying weight vectors, this allows the algorithm to track the optimal weight vector while still only incurring a finite iteration cost per frame.

A warm-start procedure cannot be used in the case of the beamformer output computation as it varies rapidly between frames. However, only a finite number of iterations are required per frame to achieve near-optimal performance. Thus, an iteration limit can be imposed to achieve a fully cyclic implementation. The performance of this iteration-limited output computation and the warm-started weight vector computation introduced above are demonstrated in Sec. 8.6.4.

8.5.8. COMPARING THE TRANSMISSION COSTS OF DIFFERENT BEAMFORMER IMPLEMENTATIONS

Table 8.1 includes the transmission costs of the distributed implementations of the BDLCMV/BDLCMP algorithm proposed in this paper. It is worth noting that these transmission costs do not include the additional overhead associated with those algorithms which exploit a TAD or the costs of forming a spanning tree. However, due to the per frequency bin nature of the algorithm, these costs are assumed to be far lower than those associated with running the algorithm.

From Table 8.1, our proposed acyclic implementation appears to require a notable increase in total transmission cost when we allow $\bar{\mathbf{P}}$ to vary. However unlike existing approaches, it does so while ensuring we exactly solve the problem in each frame. In contrast, the alternative methods listed require multiple frames to reach optimality [167]. As such, the proposed acyclic approach offers a competitive advantage as it exactly attains the performance of a centralized implementation in each frame while incurring a fixed transmission cost. In contrast, the iterative nature of DLCMV, DGSC and TI-DANSE

means that they require multiple frames to achieve the same precision, essentially scaling their effective transmission costs.

The proposed cyclic implementation of BDLCMV/BDLCMP, like other existing approaches within the literature [139, 140] allows for a tradeoff between per-frame optimality and communication overhead. Importantly, when combined with the warm-start procedure introduced in Eq. (8.13), this allows for near-optimal performance while reducing the total transmission overhead per frame. In particular, in Sec. 8.6.4 we will demonstrate the effect of combining this warm-start procedure with a single iteration, that is $t_{\max} = 1$. In this case, a negligible decrease in performance is achieved while incurring a transmission cost in line with existing acyclic distributed beamformers.

Finally, by providing two methods of beamformer output computation, we allow designers to implement a fully cyclic beamforming algorithm if they desire. Perhaps more attractive though is a hybrid style approach, similar to that used in [143], which combines cyclic weight vector computation with an acyclic output computation stage. This takes advantage of the transmission savings of both approaches while, as the acyclic topology can vary between frames, removes the need for acyclic network management in contrast to [139, 140].

8.6. EXPERIMENTAL RESULTS

We compare the performance of the proposed beamformers (except of the BDALCMV, where an estimate of $\bar{\mathbf{P}}_{\mathbf{u}}$ is difficult to obtain), and six existing centralized beamformers (the MPDR, MVDR, LCMP, LCMV, LCDS and DS) in terms of noise suppression, predicted intelligibility improvement, robustness to RATF estimation errors and TAD errors. Table 8.2 summarizes the compared linearly constrained beamformers. Note that the ALCMV and ILCMV are not included in the comparisons since there are no distributed estimation methods of p_{iso}. Note that the MPDR, MVDR, LCMP, LCMV, LCDS and DS are distributable under the distributed LCMV (DLCMV) [139], as well as the distributed DS beamformer proposed in [134]. Specifically, we examine the performance of centralized implementations of the aforementioned beamformers to which their distributed counterparts converge [139].

8.6.1. EXPERIMENT SETUP

The simulations are conducted in a simulated reverberant environment with reverberation times $T_{60} = 0.2$ s and $T_{60} = 0.5$ s using the image method [168]. A box-shaped room with dimensions $6 \times 4 \times 3$ is selected for the reverberant environment. The configuration of the nodes and acoustic sources are depicted in Fig. 8.3. We considered an example scenario where a number of people are sitting around a table with a set of mobile phones on the table, each equipped with multiple microphones. In this case, $N = 5$ nodes were placed on a virtual surface (with no physical properties) and four sources were placed around the surface. Each node was equipped with 3 microphones forming a uniform linear array with an inter-microphone distance of 2 cm. This resulted in a total of $M = 15$ microphones. Three of the four sources were interferering talkers (2 female and 1 male) with the remainder being the target source (a male talker). Each signal had a simulated duration of 30 s and was sampled at $f_s = 16$ kHz. The power of each inter-

Table 8.2: Summary of compared linearly constrained beamformers which are all special cases of the optimization problem in Eq. (8.4). Note that $\mathbf{w}^H\Lambda = \mathbf{f}^H$ is the constraints in Eq. (8.3).

Method	$\mathbf{P}$	Constraints	Target activity detection
MPDR	$\mathbf{P_y}$	$\mathbf{w}^H\mathbf{a} = 1$	no
MVDR	$\mathbf{P_n}$	$\mathbf{w}^H\mathbf{a} = 1$	yes
DS	$\mathbf{I}$	$\mathbf{w}^H\mathbf{a} = 1$	no
LCMP	$\mathbf{P_y}$	$\mathbf{w}^H\Lambda = \mathbf{f}^H$	no
LCMV	$\mathbf{P_n}$	$\mathbf{w}^H\Lambda = \mathbf{f}^H$	yes
LCDS	$\mathbf{I}$	$\mathbf{w}^H\Lambda = \mathbf{f}^H$	no
BDLCMP	$\bar{\mathbf{P}}_\mathbf{y}$	$\mathbf{w}^H\Lambda = \mathbf{f}^H$	no
BDLCMV	$\bar{\mathbf{P}}_\mathbf{n}$	$\mathbf{w}^H\Lambda = \mathbf{f}^H$	yes

ferer at its original position was set to be approximately equal to the power of the target source at its original position (i.e., a 0 dB SNR). The impulse responses between microphones and sources were computed using the toolbox in [169], with length 200 ms. The closest microphone to the target was selected as the reference microphone (see Fig. 8.3). The microphone-self noise was white Gaussian noise with 40 dB SNR with respect to the target signal at the reference microphone.

As can be noted in Fig. 8.3, the distance between any two nodes was quite big (i.e., the distance between the closest microphone-pair, where the two microphones belonged to two different nodes, was at least 0.5091 m). Thus, the ambient noise was approximately spatially uncorrelated between different nodes. As explained in Section 8.2, the late reverberation, which is the main contribution in the ambient noise component, becomes approximately uncorrelated between two microphones with distance d above a certain threshold $f_c = c/(2d)$. Here, the distance of the closest microphone-pair where the microphones belong to two different nodes is 0.5091 m corresponding to $f_c = 333.9$ Hz (if $c = 340$ m/s). Note that the correlation between any other microphone-pair with microphones in different nodes will have even smaller f_c.

On the other hand, the late reverberation for microphones within a node is highly correlated. The distance between two consecutive microphones is $d = 0.02$ m and, resulting in $f_c = 8.5$ kHz, which is greater than $f_s/2 = 8$ kHz.

8.6.2. PROCESSING

STFT frame-based beamforming was performed using an overlap and save (OLS) procedure [170]. We used a rectangular analysis window with length $2L_{\text{fr}} = 50$ ms, where $L_{\text{fr}} = 25$ ms is the length of the current frame. Thus, the early-reverberant RATF vectors of the sources are associated with an impulse response of length 50 ms. The analysis window was applied on the current frame and the previous frame in order to a) mitigate circular convolution problems, and b) to be able to handle large phase shifts in the

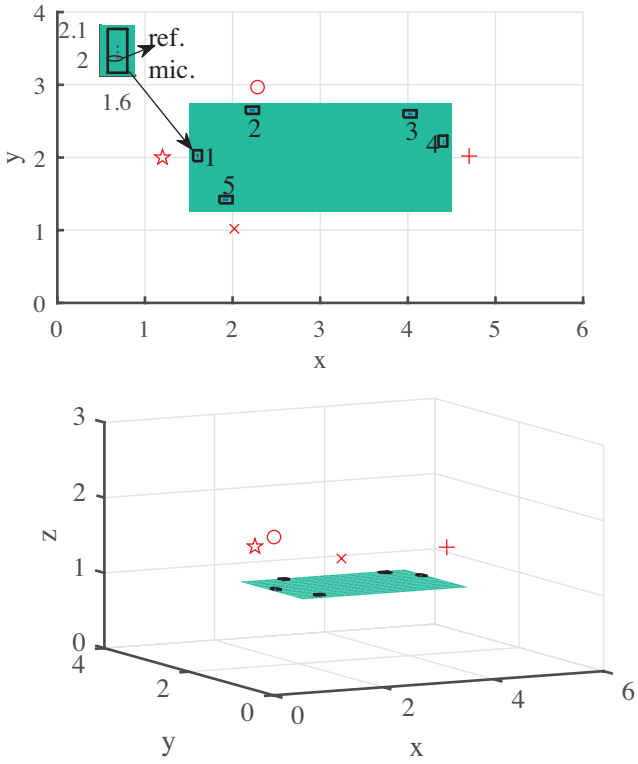


Figure 8.3: Experimental setup from two different angles: three interferers (two female talkers with markers '+' and 'x' and one male talker with marker 'o'), one target (a male talker with marker ⋆), and five nodes, with three microphones each, sitting on the virtual surface. The height of the virtual surface is 1 m.

constraints due to the large microphone array aperture. The FFT length is $\Phi = 1024$.

In order to achieve a smoother processing than standard OLS, the analysis window was shifted by $L_{\text{fr}}/2$ samples[2]. A Hann window (synthesis window) was then applied, with length L_{fr}, on the last L_{fr} processed samples. Finally, the last $L_{\text{fr}}/2$ processed samples were saved in order to add them to the corresponding samples of the next windowed segment.

The CPSDMs, for the k-th frequency bin and β-th analysis segment, were estimated via recursive averaging as described in Section 8.3.2. Note that the block-diagonal CPSDMs were recursively averaged locally at each node. The noise CPSDM and the block-diagonal noise CPSDM were estimated using an ideal TAD and a non-ideal state-of-the-art voice activity detector proposed in [171]. For simplicity, the TAD decision is based only on the reference microphone signal.

[2]The standard OLS procedure usually shifts the analysis window by L_{fr}.

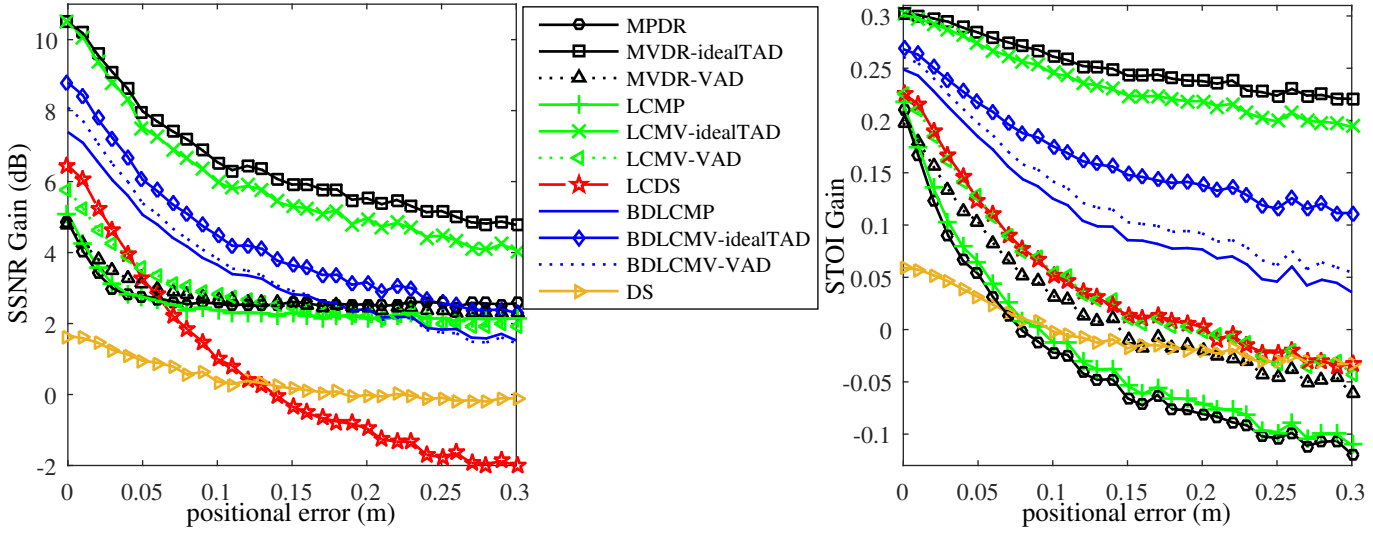


Figure 8.4: Reverberation time T_{60} = 0.2 s: Comparison of the beamformers in Table 8.2 as a function of positional error between training and testing positions. The methods that depend on a TAD are computed using an ideal TAD and the state-of-the-art voice activity detector (VAD) proposed in [171].

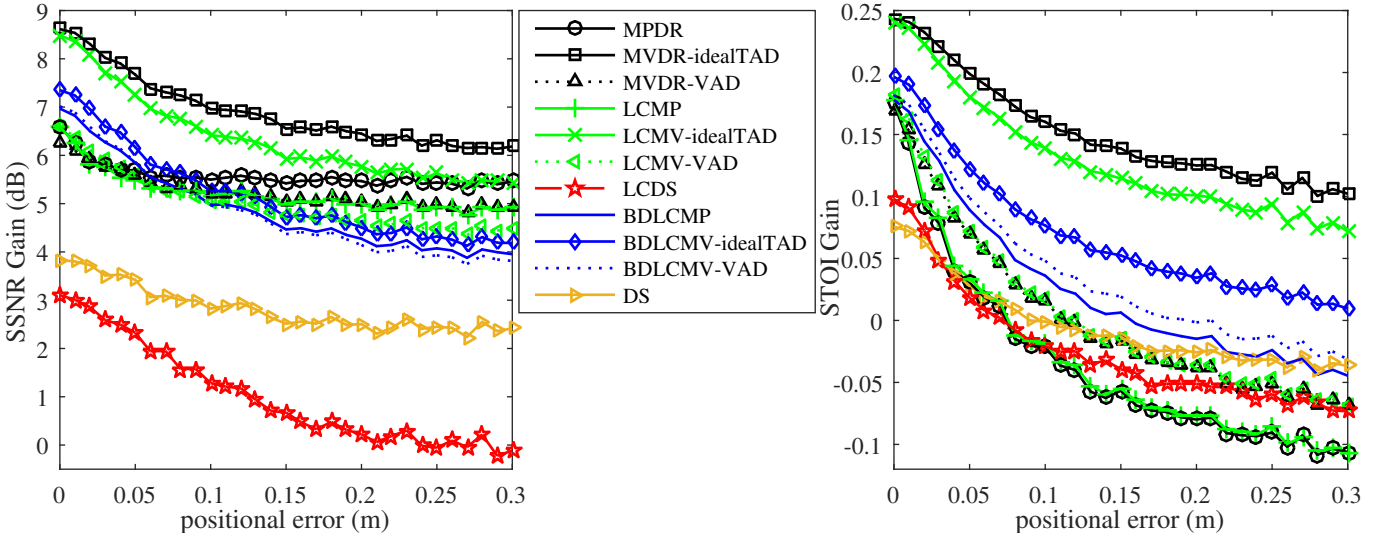


Figure 8.5: Reverberation time T_{60} = 0.5 s: Comparison of the beamformers in Table 8.2 as a function of positional error between training and testing positions. The methods that depend on a TAD are computed using an ideal TAD and the state-of-the-art voice activity detector (VAD) proposed in [171].

The RATF vectors were estimated once using additional 2 s recordings per source. Specifically, each talker spoke alone for 2 s, while all the others were silent. The CPSDM matrices of each talker were computed as described in Section 8.3.2 and the dominant relative eigenvector from each CPSDM was selected as an estimate of the RATF vector for each source[3]. These initial positions of the talkers, in which the RATF vectors were estimated, will be referred to as training positions and were nearby to the testing positions depicted in Fig. 8.3. Therefore, the RATF estimation errors of all sources can be modeled as a function of positional error between the training positions and the testing positions.

8.6.3. ROBUSTNESS TO RATF ESTIMATION ERRORS

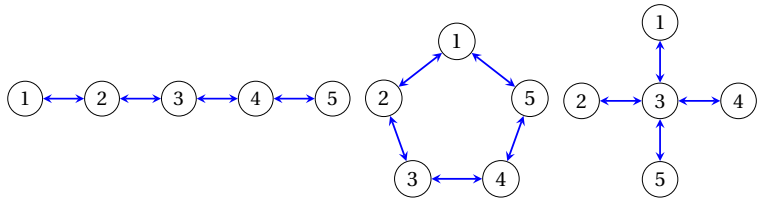
Figs. 8.4 and 8.5 show the performance of the aforementioned beamformers in terms of segmental-signal-to-noise-ratio (SSNR) gain and the short-time objective intelligibility measure (STOI) [172] gain as a function of positional error for $T_{60} = 0.2$ s and $T_{60} = 0.5$ s, respectively. Note that the noise that is computed in the SSNR consists of the interferers, background, and target distortion noise. The erroneous training locations were uniformly distributed over a sphere centered around the true source locations having a radius ranging from $0 - 0.30$ m in 0.01 m steps. For every value of positional error, the average performance of 20 different setups was measured. Each setup used the same source signals at the same testing locations as shown in Fig. 8.3. However, a different set of initial training positions, computed as mentioned previously, were used in each setup. Likewise, different realizations of the microphone-self noise were also used in each setup.

It is clear that the proposed beamformers are more robust for the combination of large positional and TAD errors. Specifically, the BDLCMV and the BDLCMP provide significantly better predicted intelligibility improvement compared to all the other methods using a non-ideal TAD or not using a TAD. The BDLCMV with the non-ideal TAD is slightly better than the BDLCMP. Thus, in this particular scenario a TAD is not necessary for the proposed method, since it will create errors and the performance advantage will be small. Note that for $T_{60} = 0.5$ s and for large positional errors, the proposed methods achieve worse noise reduction, but better intelligibility improvement, than the other methods. As explained in Section 8.5, this is because the proposed beamformers distort the target signal much less than the other beamformers.

The LCMV using the non-ideal TAD is much more robust than the LCMP and gives much higher predicted intelligibility improvement. It is worth noting that for $T_{60} = 0.2$ s the fixed LCDS has almost the same predicted intelligibility improvement as the LCMV. This makes the usage of the LCMV beamformer, in this particular acoustic scenario, obsolete in the distributed context since LCDS has significantly lower communication costs. On the other hand, for $T_{60} = 0.5$ s the performance of LCDS deteriorates significantly and becomes also worse compared to the DS beamformer. Moreover, the MVDR using a non-ideal TAD has almost the same predicted intelligibility improvement with the LCMV using the non-ideal TAD for $T_{60} = 0.5$ s.

In conclusion, for those simulations using a non-ideal TAD, the proposed methods are the most robust out of those considered. Moreover, the proposed method incurs

[3] If there is a noise component which is always active, such as an air-condition, a more accurate method of estimating the RATF of the talkers is by using the GEVD approach [157].



(a) Chain (b) Ring (c) Star

Figure 8.6: Chain, Ring and Star topologies for the considered five node network.

lower communication costs, as explained in Section 8.5, making it a strong candidate for distributed beamforming.

8.6.4. LIMITING ITERATIONS PER FRAME FOR PDMM BASED BDLCMP/BDLCMV

We now compare the impact of a finite iteration cap on the optimality of both the computed beamformer weight vector and beamformer output signal. For these simulations, the same setup, as introduced in Sec. 8.6.1, was used. The case of BDLCMP with no RATF estimation errors was considered where by the centralized beamformers used previously were substituted with their cyclic counterparts introduced in Sec. 8.5.5. For these simulations, three standard network configurations (a chain, a ring and a star network) were considered to highlight the impact network topology can play on convergence. Examples of these three network topologies are included below in Figures 8.6a, 8.6b, 8.6c respectively. A step size of $\rho = \frac{1}{2}$ was heuristically selected for all simulations. With a more refined selection of this parameter, we expect that faster convergence could be achieved.

Fig. 8.7 shows a comparison of convergence rates of both cold and warm-started beamformer weight vector computation for the three networks considered. As expected, while all three methods require many iterations (> 30) to achieve reasonable weight vector estimation, when combined with a warm-start procedure, even a single iteration per frame achieves near optimal gains in both STOI and SSNR. Thus, for slowly varying CPSDM estimates, the cyclic BDLCMP/BDLCMV approach offers an opportunity to dramatically reduce transmission costs while maintaining near optimal performance. Furthermore, the effectiveness of this warm-start does not seem to vary significantly with network topology.

For beamformer output computation, as demonstrated in Fig. 8.8, the story is similar. While the dynamic nature of the beamformer output does not facilitate a warm-start procedure, the simplicity of the problem means that within 10 iterations or so a near optimal beamformer output is computed.

Unlike the beamformer weight vector computation, here we can more clearly observe the effect of network topology on convergence. In particular, the chain network, which has a larger diameter than either the ring or the star network, requires roughly twice the number of iterations to approach optimal convergence. This point is consistent with the fact that an even length chain network has twice the diameter of a ring

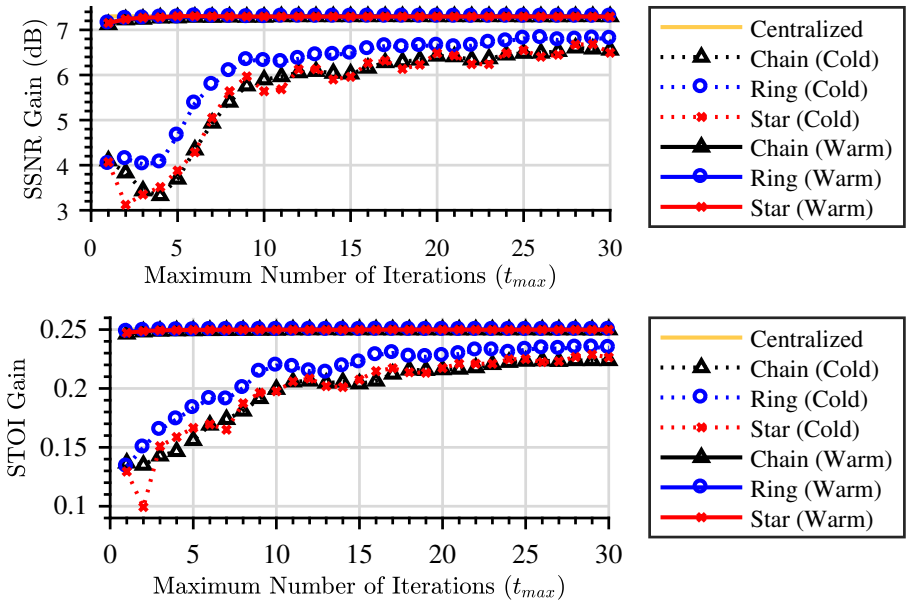


Figure 8.7: Comparing the effect of a finite iteration limit on PDMM beamformer weight vector computation. Cold-start (cold) and warm-start (warm) scenarios are considered with the beamformer output being computed exactly via acyclic data aggregation.

network of the same size. However, this may be able to be remedied with more careful step size selection.

8.7. CONCLUSION

In this paper, we proposed a new distributed linearly constrained beamformer, which provides increased robustness to TAD and RATF estimation errors compared to traditional LCMV-based beamformers. Moreover, the proposed approach is immediately distributable due to its use of a block-diagonal CPSDM. Unlike most competing distributed beamformers, the proposed method can be applied in arbitrary network topologies, while at the same time having much lower communication costs in comparison to competing cyclic approaches and comparable costs to acyclic ones. Furthermore, the general nature of the distributed algorithm facilitates a trade off between transmission costs and per-frame optimality allowing it to be tailored to the needs of a particular application.

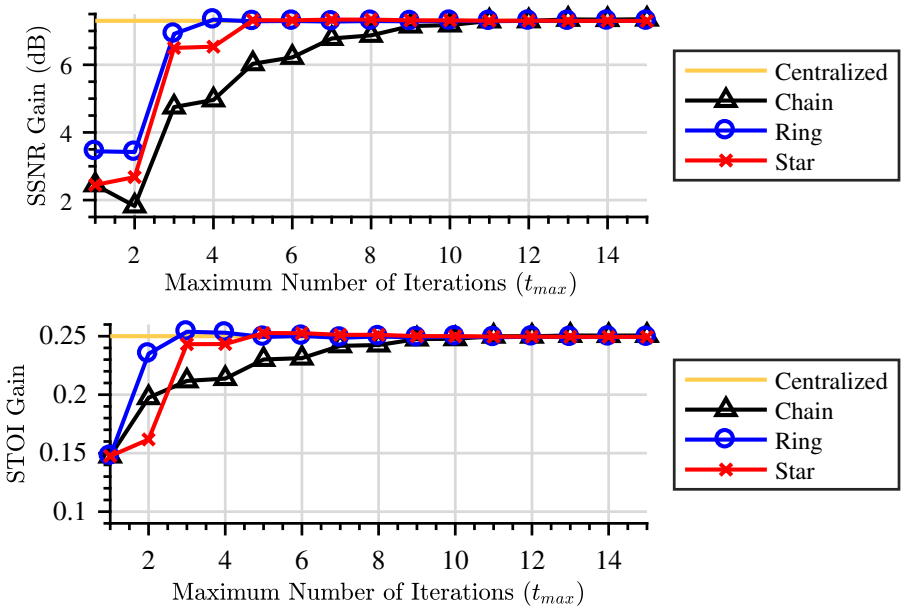


Figure 8.8: Comparing the effect of a finite iteration limit on PDMM beamformer output computation. For each of the networks considered the beamformer weight vector is computed exactly via acyclic data aggregation.

V

EPILOGUE

9

CONCLUSIONS AND FUTURE WORK

"They were close to the end of the beginning . . ."

Stephen King - *The Gunslinger*

9.1. CONCLUSIONS

Distributed convex optimization provides a flexible basis for performing signal processing in computational networks, particularly those that lack structure and hierarchy. For applications such as those involving IoT devices, adhoc sensor networks and more, such approaches represent an opportunity to capitalize on the large volumes of data being generated as we transition towards a massively networked world. In support of this, in this thesis we have used monotone operator theory as a tool to both expand our understanding of existing solvers within the literature, as well as to propose new approaches to broaden the scope of what can be optimized in a distributed fashion. Our contributions to these general goals are summarized below.

9.1.1. ANALYSIS OF EXISTING DISTRIBUTED SOLVERS

In the area of solver analysis, we demonstrated how the primal dual method of multipliers (PDMM) can be derived from a monotone operator theoretic perspective, linking its convergence characteristics with the well known Peaceman-Rachford (PR) operator splitting approach. The importance of this link is that, unlike existing derivations from within the literature, the monotone operator perspective adopted allowed us to demonstrate firstly how strong convexity and differentiability were sufficient conditions to guarantee convergence to a primal optimal solution and furthermore that under the additional assumption of Lipschitz continuity, that this convergence would occur at a geometric rate. Specifically, we were able to show how, in the case of consensus problems, this rate was dependent on the underlying topology of the network and thus to provide a framework to quantify the performance of a range of deterministic network topologies. Additionally, we provided example problems for which PDMM would not converge, an important point currently missing from the literature. While a drawback for the generality of the use of PDMM, using our new insight into the operation of the algorithm from a monotone perspective, we were able to propose a simple modification to address this via the inclusion of additional primal regularization.

9.1.2. DISTRIBUTED SOLVER DESIGN

In complement to our research on the analysis of PDMM, in Chapters 6 and 7, our objective was to broaden the classes of problems which we could solve in a distributed manner. Our first efforts in this direction highlighted how problem separability was a sufficient property to guarantee distributability of a problem. Using this point we introduced a novel algorithm called the distributed method of multipliers (DMM) to solve such problems. Specifically, in Chapter 6 we showed how DMM could be formed via monotone operator theory by combining a particular lifted dual form with classic averaged Peaceman-Rachford iterations. Our second contribution in this area focused on the design of a novel solver for distributed consensus in time-varying networks. Using a particular time-varying metric choice, in Chapter 7 we demonstrated how classic monotone operator theory could be combined with a certain iterate reparameterization procedure to produce an algorithm somewhat akin to that seen in Push-Sum type methods [11, 40], i.e. being comprised of local computations at each node followed by an averaging step using a certain weighted graph Laplacian. We further demonstrated how the proposed method exhibited guaranteed convergence for strongly convex objective functions as

well as highlighting its empirical performance in the case of more general closed, convex and proper functions. Overall, the proposed methods both address open problem areas within the literature and complement existing efforts to broaden the applicability of distributed optimization solvers.

9.1.3. PRACTICAL DISTRIBUTED CONVEX OPTIMIZATION

In addition to the more fundamental contributions, we demonstrated the use of PDMM in the practical signal processing task of multichannel speech enhancement in wireless acoustic sensor networks (WASN). In particular, we demonstrated how a particular modeling of the acoustic scene leads to a separable equivalent optimization problem to which PDMM can be naturally applied. This particular model had the additional benefit of providing increased robustness to steering vector mismatch which typically plagues more traditional beamforming methods. Furthermore, we explored the practical considerations which must be traded off in the context of processing in WASNs such as the necessity for deterministic signal processing, i.e., signal processing in a predefined time window. Such practical considerations represent an area of distributed signal processing not directly handled by distributed convex optimization. This practical example therefore also highlighted the importance of both understanding the underlying target problem as well as the distributed optimization techniques required to produce algorithms for deployment in real world systems.

9.2. FUTURE RESEARCH

In addition to the initial questions which motivated the work of this thesis, a number of complementary topics presented themselves over the course our research. To conclude this dissertation, we therefore present a selection of potential directions for future research which fall into the area of distributed convex optimization.

9.2.1. ASYNCHRONOUS DISTRIBUTED OPTIMIZATION

One of the biggest limitations of the research undertaken thus far is that we assume that computation across multiple nodes can be synchronized. While this could be addressed through the use of some form of additional clock synchronization technique, such an overhead is an undesirable addition, particularly for adhoc systems. In a similar vein, in heterogenous problems, i.e., where the local objectives of each node are different, some nodes may require move time to compute local updates. In such a context a synchronized updating scheme is dependent on the needs of the slowest node. A more desirable system model addressing both of these issues would be to allow the nodes in the network to update independently of one another, while again achieving the same node based computation and edge based communication as the synchronous methods presented in this work.

In the case of PDMM, asynchronous operation can be trivially implemented due to its inherently node based structure. Our initial empirical results in this direction indicate stronger convergence results than in the synchronous context. In particular, the types of problems for which synchronous PDMM may not converge, such as problems objective functions lack strong convexity and differentiability, exhibit convergence when

solved asynchronously in our preliminary simulations. This leads to the following research questions.

Question 7. *Is node asynchronicity a sufficient condition for the convergence of PDMM for CCP functions? If so how does this asynchronous operation affect existing convergence rate bounds for stronger function classes such as those which exhibit strong convexity and smoothness?*

9.2.2. OPTIMIZATION IN DIRECTED NETWORKS

In addition to the challenge of asynchronous updating schemes, a second limitation of this thesis is the assumption that the underlying networks can be modeled using undirected graphs. In particular, for a heterogenous network topology, i.e., one where the nodes of the network may exhibit different physical architectures, it may be that two way communication cannot be guaranteed between all pairs of nodes in the network. For instance, if one node has a greater transmission power than the other, a uni-directional communication link would establish itself. This would in turn result in a directed graph topology. While there are methods within the literature that support such a directed network structure, such as the Push-Sum family of algorithms [39], it would be interesting to see if the methods proposed in this work could be generalized to such a context. This therefore raises the following research question.

Question 8. *How can we design distributed optimization solvers for use in directed networks? In particular, can we use monotone operator theory to facilitate this process?*

9.2.3. QUANTIZATION EFFECTS IN DISTRIBUTED OPTIMIZATION

Another consideration of practical systems is that data shared between nodes will be encoded using some form of finite precision. Quantization effects within distributed algorithms have received reasonable attention within the optimization literature with inexact Krasnosel'skiĭ-Mann iterations [173] representing one way to analyze these affects. However, the question of distributed quantizer design, how one can design quantizers for the specific task of distributed optimization, is still an open area of research. For instance, some of our preliminary work demonstrated how dynamic one-bit quantization could be combined with PDMM to yield guaranteed convergence to optimal solutions while offering a significant reduction in bit rate for certain problem classes [174]. This raises the more general research question:

Question 9. *How can we design quantization schemes for use in distributed signal processing applications without compromising convergence? If so, how do these methods affect algorithmic convergence and overall data transmission?*

9.2.4. DISTRIBUTED NON-CONVEX OPTIMIZATION

A more general branch of additional research and one which we did not touch on in this work, is the design of distributed solvers for non-convex optimization problems. In particular, while the world of distributed convex optimization has seen a reasonable treatment within the literature, solving non-convex problems, if even only to a stationary point, has not seen such an extensive investigation. It would be interesting to see if

the monotone operator theory perspective can be applied in this context to yield convergence results, even if one to local stationary points. In particular, in the case of fields such as machine learning and deep learning, non-convex optimization and their associated solvers form an essential component of network training and so devising efficient distributed implementations for this context could have appealing implications for a large family of real world tasks.

Question 10. *Can monotone operator theory be used to design distributed solvers for non-convex optimization?*

9.2.5. ACCELERATED SOLVER DESIGN

Finally, one of the inherent drawbacks of distributed optimization is its iterative nature. As we have shown in our work, this iterative nature can lead to slow convergence rates particularly for ill-chosen network topologies. This problem is also reflected across many commonly used convex solvers and thus algorithm acceleration has become a strong focus for a number of methods. An interesting question to ask is whether such acceleration procedures can also be applied in the case of distributed optimization to improve convergence. One such area of focus for these methods could be through the use of quasi-newton type methods where by previous iterates are stored locally at each node which can then be used to construct more informed updates in turn. Again, improving distributed solver time can only increase their applicability to real world tasks and thus could offer a significant benefit to the world of distributed signal processing. A particular research question which therefore sparked our attention was the following.

Question 11. *How can we accelerate the convergence rates of distributed solvers through the use of on node memory?*

9.3. CLOSING REMARKS

The work summarized in this thesis represents our first foray into the use of monotone operator theory in the world of distributed convex optimization. Providing a mathematical basis for the treatment of nonlinear solver design in networks, this perspective facilitated the analysis of a variety of approaches in a systematic format. Importantly, this connection has lead to the development of both a better understanding of existing distributed methods as well as the design of new approaches in turn. While only a drop in the ocean of the knowledge on distributed signal processing, we therefore hope that the contributions in this work, as well as the additional branches of research proposed, have been engaging for the reader and perhaps even stimulated an interest in distributed signal processing in turn.

SUMMARY

Collaboration - The act of working together to achieve a common goal.

FOLLOWING their conception in the mid twentieth century, the world of computers has evolved from a landscape of isolated entities into a sprawling web of interconnected machines. Yet, given this evolution, many of the methods we use for allowing computers to work together still reflect their inherently isolated origins with the aggregation of data or master-slave relationships still commonly seeing use. While sufficient for some types of applications, these approaches do not naturally reflect the collaboration strategies we observe in nature and so the question is raised as to whether we can do better?

In parallel to the improvements in computer to computer communication, the emergence of new paradigms such as the Internet of Things (IoT), Big Data processing and cloud computing in recent years has placed an increasing importance on networked systems in many facets of the modern world. From power grid management, to autonomous vehicle navigation, to even our basic means of interaction through social media, these networks are a pervasive presence in our day to day lives. The vast amounts of data generated by these networks and their ever increasing sizes makes it impractical if not impossible to resort to traditional centralized processing and therefore necessitates the search for new methods of signal processing within networked systems.

In this thesis we approach the task of distributed signal processing by exploiting the synergy between such tasks and equivalent convex optimization problems. Specifically, we focus on the task of distributed convex optimization, that of solving optimization problems involving groups of computers in a collaborative manner and the development of distributed solvers for such tasks. Such solvers distinguish themselves by only allowing local computations at each computer in a network and the exchange of information between connected computers. In this way, distributed solvers naturally respect the structure of the underlying network in which they are deployed.

In the pursuit of our goal, we approach the task of distributed solver design via the lens of monotone operator theory. Providing a well known platform for the derivation of many first order convex solvers, herein we demonstrate the use of this theory as a means of constructing and analyzing a number of algorithms for distributed optimization. The first major contribution of this thesis lies in the analysis and understanding of an existing algorithm for distributed optimization within the literature termed the primal dual method of multipliers (PDMM). In particular, by demonstrating a novel interpretation of PDMM from the perspective of monotone operator theory we are able to better understand its convergent characteristics and highlight sufficient conditions for which PDMM will converge at a geometric rate. Furthermore we quantify the impact that network topology has on these convergence rates, drawing a direct connection between spectral characteristics of networks and distributed optimization.

Secondly, we explored the space of solver design by proposing novel algorithms for distributed networks. For the family of separable optimization problems, those with separable objectives and constraints, we demonstrated a distributed solver design using a specific lifted dual form. Based on monotone operator theory, the convergence analysis of the proposed method followed naturally from well known results and broadened the class of distributable problems compared to the likes of PDMM. Furthermore, in the case of time-varying consensus problems, we again proposed a new algorithm by combining a network dependent metric choice with classic operator splitting methods. Again the monotone basis of this algorithm facilitated the convergence analysis of this method which empirically was also shown to converge for general closed, convex and proper functions.

Finally, we demonstrated how these methods could be used for practical distributed signal processing in networks by considering the case of multichannel speech enhancement in wireless acoustic sensor networks. By combining a particular modeling of the acoustic scene with the algorithms mentioned above, the proposed method was not only distributable but also offered increased resilience to steering vector mismatch than other standard approaches. This example also highlights the importance of understanding both the target application and the distributed solvers themselves in developing effective solutions.

Overall, this thesis provides a first foray into the world of distributed optimization via the lens of monotone operator theory. We feel that this perspective provides an ideal reference for the analysis of such algorithms while also providing a general framework for convex optimization solver design in turn. While this thesis is not the end of this branch of research, it indicates the potential of the monotone operator theory as a unifying method for the development and analysis of distributed optimization solutions.

SAMENVATTING

VANAF de introductie van de computer halverwege de twintigste eeuw, is de wereld van computers veranderd van een landschap van geïsoleerde entiteiten tot een uitgestrekt web van onderling verbonden computers. Desalniettemin, gegeven deze evolutie, zijn de technieken die we gebruiken om computers te laten samenwerken nog steeds gebaseerd op de oorspronkelijke situatie waarbij het verzamelen van data en het master-slave principe veelvuldig voorkomt. Hoewel dit principe volstaat voor sommige toepassingen, reflecteert deze aanpak niet de samenwerkingsstrategieën die we in de natuur tegenkomen en rijst de vraag of we computers op een betere manier kunnen laten samenwerken.

Naast de verbeterde communicatie tussen computers heeft de opkomst van nieuwe paradigma's zoals het Internet-of-things (IoT), Big Data verweking en cloud-computing er voor gezorgd dat vele facetten van netwerksystemen een steeds belangrijker rol hebben gekregen. Zo zijn power-grid management, navigatie van autonome voertuigen, en zelfs onze manier van interactie via sociale media, tegenwoordig niet meer weg te denken uit ons dagelijks leven. De enorme hoeveelheden gegevens die door deze netwerken worden gegenereerd maken het onpraktisch, zo niet onmogelijk, om op de traditionele, gecentraliseerde manier verwerkt te worden en vereist daarom nieuwe methoden voor signaalverwerking binnen netwerksystemen.

In dit proefschrift benaderen we het probleem van gedistribueerde signaalverwerking door gebruik te maken van de synergie die dit probleem heeft met equivalente convexe optimalisatieproblemen. We richten ons in het bijzonder op de taak van gedistribueerde convexe optimalisatie, het oplossen van optimalisatieproblemen met behulp van verschillende samenwerkende computers, en de ontwikkeling van gedistribueerde methoden om dergelijke taken op te lossen. Dergelijke methoden onderscheiden zich door het uitwisseling van informatie tussen met elkaar verbonden computers en het feit dat alleen lokale berekeningen op de computers in het netwerk zijn toegestaan. Op deze manier maken de gedistribueerde methoden op een natuurlijke manier gebruik van de structuur van het onderliggende netwerk.

Om ons doel te bereiken benaderen we het probleem van het ontwerp van gedistribueerde methoden door gebruik te maken van de theorie van monotone operatoren. Gegeven een bekend platform voor het ontwerp van eerste-orde convexe methoden, laten we in dit proefschrift zien dat deze theorie ook goed gebruikt kan worden voor het analyseren en ontwikkelen van een aantal algoritmen voor gedistribueerde optimalisatie. De eerste belangrijke contributie van dit proefschrift bestaat uit het analyseren en begrijpen van een bestaand algoritme voor gedistribueerde optimalisatie, welke in de literatuur bekend is onder de naam primal-dual method of multipliers (PDMM). Door een nieuwe interpretatie van het PDMM algoritme zijn we in staat om het convergentiegedrag beter te begrijpen en geven we voldoende voorwaarden voor lineaire convergentie. Daarnaast kwantificeren we de invloed van de netwerktopologie op de conver-

gentiesnelheid door een directe link te leggen tussen spectrale eigenschappen en gedistribueerde optimalisatie.

De tweede bijdrage van dit proefschrift is het exploreren van ontwerpvrijheden door nieuwe algoritmen te introduceren voor gedistribueerde netwerken. We demonstreren een gedistribueerde methode voor de klasse van scheidbare optimalisatieproblemen, problemen waarbij zowel de kostfunctie als de randvoorwaarden scheidbaar zijn, door gebruik te maken van een specifieke uitbreiding van de duale probleembeschrijving. Door gebruik te maken van monotone operatortheorie volgt de convergentieanalyse op een natuurlijke manier uit bestaande resultaten en zorgt daarmee voor een uitbreiding van de klasse van gedistribueerde methoden die vergelijkbaar zijn met het PDMM algoritme. Daarnaast introduceren we een algoritme dat gebruikt kan worden om tijdvariërende consensus problemen op te lossen door een netwerk afhankelijk maat in te voeren en deze te combineren met klassieke operator scheidingstechnieken. Ook hier faciliteert de monotone operatoreigenschap de convergentieanalyse waarbij we tevens, hetzij empirisch, laten zien dat het algoritme voor algemene gesloten, convexe en eigenlijke functies convergeert.

Tot slot laten we zien hoe dergelijke methoden gebruikt kunnen worden in een praktische toepassing van meerkanaals spraakverbetering in draadloze akoestische sensornetwerken. Door de akoestisch omgeving op een bepaalde manier te modeleren, en deze te combineren met de hierboven genoemde algoritmes, laten we zien dat het probleem niet alleen gedistribueerd opgelost kan worden, maar dat deze aanpak ook leidt tot een zekere ongevoeligheid voor fouten in de akoestische overdrachtsfunctie. Dit voorbeeld toont eens temeer aan dat het noodzakelijk is om kennis te hebben van zowel de applicatie als gedistribueerde methoden om dergelijke problemen efficiënt op te lossen.

Samenvattend geeft dit proefschrift een eerste kennismaking met de wereld van gedistribueerde optimalisatie bekeken door de lens van monotone operatortheorie. We zijn van mening dat dit perspectief een ideale referentie biedt voor de analyse van dergelijke algoritmen en tegelijkertijd een algemeen kader schetst voor convex optimalisatie methoden. Hoewel dit proefschrift niet het einde van deze tak van onderzoek is, toont dit werk het potentieel van monotone operatortheorie als een overkoepelende methode voor de ontwikkeling en analyse van gedistribueerde optimalisatiemethoden.

ACKNOWLEDGEMENTS

"Well, we knocked the bastard off."

Sir Edmund Hillary

This thesis, while a byproduct of my studies over the last four years, is certainly not something I have achieved on my own. While I cannot name everybody here, I would like to offer my sincerest thanks to a number of people.

Firstly, to my promotors Bastiaan and Richard, your guidance these past four years has been unending and your patience unwavering. There were many times when I could not see the wood for the trees, when I felt lost and overwhelmed and you were always solid references. Thank you for letting me explore when I could and for nudging me back on track when I strayed to far off course. In particular, to Bastiaan I am so grateful for the faith you put in me by offering me the chance to come to Delft. I am so lucky to have had the chance to have you both guiding me through this time.

To my office mates, Andreas, Aydin, Elvin, Jamal, Jie, Pim and Wangyang, thanks for putting up with me for these last four years. Through it all, the laughs, the sport, the cake, the camaraderie, the philosophy and the jokes, I feel that I have grown so much during my time with you. To my other colleagues, whether it be running, football, volleyball, rock climbing, cross country skiing, exploring, dinner, coffee or more, thank you so much for making my years in Delft enjoyable and rememberable. I am so happy to have been a part of the quirky family we have at CAS.

To my friends, where ever you are in the world, be it New Zealand, the Netherlands, or even further afield, thank you for being who you are. I know that I am not the best at keeping in touch but when we do meet up I cherish every moment. It is hard living between both sides of the world but at the same time it makes the times we have shared and will continue to share in the future that much more special.

To my family, my mother Mandy, my father Glenn and my brother Ben, I am sorry I have been gone for so long and I am sorry that I will be gone for a while longer. Being so far away from you for so long has been hard and I am so grateful for your constant love and support. While I hope that one day we can live closer together again, I am glad that we live in an age where Skype exists and we can talk in a way that makes the distance feel a bit less, even if only for a moment. You mean the world to me and I miss you every day.

Finally, to Elke. I don't think I would be the person I am now if we had not met. Thank you for picking me up when things were too tough, for reveling in the good times when things were good and for always pushing me to be the best that I can. Through all the mountains we have climbed, and valleys we have walked, thank you for just being you. I love you and I can't wait to see what the future for us together holds.

BIBLIOGRAPHY

[1] R. Hanna, A. Rohm, and V. L. Crittenden, "We're all connected: The power of the social media ecosystem," *Business horizons*, vol. 54, no. 3, pp. 265–273, 2011.

[2] N. Antonopoulos and L. Gillam, *Cloud computing*. Springer, 2010.

[3] P. Palensky and D. Dietrich, "Demand side management: Demand response, intelligent energy systems, and smart loads," *IEEE transactions on industrial informatics*, vol. 7, no. 3, pp. 381–388, 2011.

[4] M. Gerla, E.-K. Lee, G. Pau, and U. Lee, "Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds," in *Internet of Things (WF-IoT), 2014 IEEE World Forum on*. IEEE, 2014, pp. 241–246.

[5] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[6] T. Danova, "Morgan stanley: 75 billion devices will be connected to the internet of things by 2020," *Business Insider*, vol. 2, 2013.

[7] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things journal*, vol. 1, no. 1, pp. 22–32, 2014.

[8] H. Messer, "Wireless communication links as opportunisic iot for near ground rain monitoring," in *2018 IEEE Statistical Signal Processing Workshop (SSP)*. IEEE, 2018, pp. 115–119.

[9] P. Kumar, L. Morawska, C. Martani, G. Biskos, M. Neophytou, S. Di Sabatino, M. Bell, L. Norford, and R. Britter, "The rise of low-cost sensing for managing air pollution in cities," *Environment international*, vol. 75, pp. 199–205, 2015.

[10] A. Nedic, A. Olshevsky, A. Ozdaglar, and J. Tsitsiklis, "On distributed averaging algorithms and quantization effects," *IEEE Trans. on Automatic Control*, vol. 54, no. 11, pp. 2506–2517, 2009.

[11] A. Nedić and A. Olshevsky, "Distributed optimization over time-varying directed graphs," *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 601–615, 2015.

[12] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE/ACM Trans. on Networking (TON)*, vol. 14, no. SI, pp. 2508–2530, 2006.

[13] Y. Weiss and W. Freeman, "On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs," *IEEE Trans. on Information Theory*, vol. 47, no. 2, pp. 736–744, 2001.

[14] A. Schwing, T. Hazan, M. Pollefeys, and R. Urtasun, "Distributed message passing for large scale graphical models," in *Computer vision and pattern recognition (CVPR), 2011 IEEE Conf. on*. IEEE, 2011, pp. 1833–1840.

[15] K. Murphy, Y. Weiss, and M. Jordan, "Loopy belief propagation for approximate inference: An empirical study," in *Proceedings of the Fifteenth Conf. on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1999, pp. 467–475.

[16] S. Segarra, A. G. Marques, and A. Ribeiro, "Distributed implementation of linear network operators using graph filters," in *Communication, Control, and Computing (Allerton), 2015 53rd Annual Allerton Conference on*. IEEE, 2015, pp. 1406–1413.

[17] A. Loukas, A. Simonetto, and G. Leus, "Distributed autoregressive moving average graph filters," *IEEE Signal Processing Letters*, vol. 22, no. 11, pp. 1931–1935, 2015.

[18] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[19] P. Bianchi, W. Hachem, and I. Franck, "A stochastic coordinate descent primal-dual algorithm and applications," in *Machine Learning for Signal Processing (MLSP), 2014 IEEE Int. Workshop on*. IEEE, 2014, pp. 1–6.

[20] P. Latafat and P. Patrinos, "Asymmetric forward-backward-adjoint splitting for solving monotone inclusions involving three operators," *Computational Opt. and Applications*, pp. 1–37, 2016.

[21] Y. Nesterov and A. Nemirovskii, *Interior-point polynomial algorithms in convex programming*. Siam, 1994, vol. 13.

[22] D. P. Palomar and Y. C. Eldar, *Convex optimization in signal processing and communications*. Cambridge university press, 2010.

[23] A. B. Gershman, N. D. Sidiropoulos, S. Shahbazpanahi, M. Bengtsson, and B. Ottersten, "Convex optimization-based beamforming," *IEEE Signal Processing Magazine*, vol. 27, no. 3, pp. 62–75, 2010.

[24] G. E. Dullerud and F. Paganini, *A course in robust control theory: a convex approach*. Springer Science & Business Media, 2013, vol. 36.

[25] M. Zibulevsky and M. Elad, "L1-l2 optimization in signal and image processing," *IEEE Signal Processing Magazine*, vol. 27, no. 3, pp. 76–88, 2010.

[26] R. Rockafellar, *Convex analysis*. Princeton, NJ: Princeton University Press, 1970.

[27] ——, "Monotone operators and the proximal point algorithm," *SIAM journal on control and Opt.*, vol. 14, no. 5, pp. 877–898, 1976.

[28] ——, "On the maximal monotonicity of subdifferential mappings," *Pacific Journal of Mathematics*, vol. 33, no. 1, pp. 209–216, 1970.

[29] D. Bertsekas and J. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Prentice hall Englewood Cliffs, NJ, 1989, vol. 23.

[30] J. Tsitsiklis, "Problems in decentralized decision making and computation." Massachusetts Inst of Tech Cambridge lab for information and decision systems, Tech. Rep., 1984.

[31] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Trans. on automatic control*, vol. 31, no. 9, pp. 803–812, 1986.

[32] J. Eckstein, "Splitting methods for monotone operators with applications to parallel optimization," Ph.D. dissertation, Massachusetts Institute of Technology, 1989.

[33] ——, "Parallel alternating direction multiplier decomposition of convex programs," *Journal of Opt. Theory and Applications*, vol. 80, no. 1, pp. 39–62, 1994.

[34] H. Bauschke and P. Combettes, *Convex analysis and monotone operator theory in Hilbert spaces*. New York, NY.: Springer NY, 2017, vol. 408.

[35] J. C. Nash, "The (dantzig) simplex method for linear programming," *Computing in Science & Engineering*, vol. 2, no. 1, pp. 29–31, 2000.

[36] L. G. Khachiyan, "A polynomial algorithm in linear programming," in *Doklady Academii Nauk SSSR*, vol. 244, 1979, pp. 1093–1096.

[37] M. Zhu and S. Martínez, "On distributed convex optimization under inequality and equality constraints," *IEEE Trans. on Automatic Control*, vol. 57, no. 1, pp. 151–164, 2012.

[38] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[39] K. I. Tsianos, S. Lawlor, and M. G. Rabbat, "Consensus-based distributed optimization: Practical issues and applications in large-scale machine learning," in *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*. IEEE, 2012, pp. 1543–1550.

[40] ——, "Push-sum distributed dual averaging for convex optimization," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*. IEEE, 2012, pp. 5453–5458.

[41] A. Chambolle, V. Caselles, D. Cremers, M. Novaga, and T. Pock, "An introduction to total variation for image analysis," *Theoretical foundations and numerical methods for sparse recovery*, vol. 9, no. 263-340, p. 227, 2010.

[42] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 2, pp. 210–227, 2009.

[43] D. L. Donoho, "Compressed sensing," *IEEE Transactions on information theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[44] R. G. Baraniuk, "Compressive sensing [lecture notes]," *IEEE signal processing magazine*, vol. 24, no. 4, pp. 118–121, 2007.

[45] S. Mallat, *A wavelet tour of signal processing: the sparse way*. Academic press, 2008.

[46] D. Malioutov, M. Cetin, and A. S. Willsky, "A sparse signal reconstruction perspective for source localization with sensor arrays," *IEEE transactions on signal processing*, vol. 53, no. 8, pp. 3010–3022, 2005.

[47] E. Ryu and S. Boyd, "Primer on monotone operator methods," *Appl. Comput. Math*, vol. 15, no. 1, pp. 3–43, 2016.

[48] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[49] D. O?Connor and L. Vandenberghe, "On the equivalence of the primal-dual hybrid gradient method and douglas-rachford splitting," 2017.

[50] G. Zhang and R. Heusdens, "Distributed optimization using the primal-dual method of multipliers," *IEEE Trans. on Signal and Information Processing over Networks*, 2016, accepted for publication.

[51] ——, "On simplifying the primal-dual method of multipliers," in *2016 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 4826–4830.

[52] A. Simonetto and H. Jamali-Rad, "Primal recovery from consensus-based dual decomposition for distributed convex optimization," *Journal of Opt. Theory and Applications*, vol. 168, no. 1, pp. 172–197, 2016.

[53] D. Mosk-Aoyama, T. Roughgarden, and D. Shah, "Fully distributed algorithms for convex optimization problems," *SIAM Journal on Opt.*, vol. 20, no. 6, pp. 3260–3279, 2010.

[54] D. Mateos-Núnez and J. Cortés, "Distributed subgradient methods for saddle-point problems," in *Decision and Control (CDC), 2015 IEEE 54th Annual Conference on*. IEEE, 2015, pp. 5462–5467.

[55] T.-H. Chang, A. Nedić, and A. Scaglione, "Distributed constrained optimization by consensus-based primal-dual perturbation method," *IEEE Trans. on Automatic Control*, vol. 59, no. 6, pp. 1524–1538, 2014.

[56] T.-H. Chang, "A proximal dual consensus admm method for multi-agent constrained optimization," *IEEE Trans. on Signal Processing*, vol. 64, no. 14, pp. 3719–3734, 2016.

[57] S. Lee and M. M. Zavlanos, "Distributed primal-dual methods for online constrained optimization," in *American Control Conference (ACC), 2016*. IEEE, 2016, pp. 7171–7176.

[58] I. Notarnicola and G. Notarstefano, "Constraint coupled distributed optimization: Relaxation and duality approach," *arXiv preprint arXiv:1711.09221*, 2017.

[59] A. Falsone, K. Margellos, S. Garatti, and M. Prandini, "Dual decomposition for multi-agent distributed optimization with coupling constraints," *Automatica*, vol. 84, pp. 149–158, 2017.

[60] A. Nedić and A. Olshevsky, "Stochastic gradient-push for strongly convex functions on time-varying directed graphs," *IEEE Transactions on Automatic Control*, vol. 61, no. 12, pp. 3936–3947, 2016.

[61] B. Gerencsér and J. M. Hendrickx, "Push sum with transmission failures," *IEEE Transactions on Automatic Control*, 2018.

[62] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the world with wireless sensor networks," in *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE Int. Conf. on*, vol. 4. IEEE, 2001, pp. 2033–2036.

[63] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer networks*, vol. 38, no. 4, pp. 393–422, 2002.

[64] A. Swami, Q. Zhao, Y.-W. Hong, and L. Tong, *Wireless sensor networks: Signal processing and communications*. John Wiley & Sons, 2007.

[65] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proceedings of the 1st ACM Int. workshop on Wireless sensor networks and applications*. Acm, 2002, pp. 88–97.

[66] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao, "Habitat monitoring: Application driver for wireless communications technology," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 2 supplement, pp. 20–41, 2001.

[67] V. C. Gungor, B. Lu, and G. P. Hancke, "Opportunities and challenges of wireless sensor networks in smart grid," *IEEE Trans. on industrial electronics*, vol. 57, no. 10, pp. 3557–3564, 2010.

[68] F. Blaabjerg, R. Teodorescu, M. Liserre, and A. Timbus, "Overview of control and grid synchronization for distributed power generation systems," *IEEE Trans. Industrial Electronics*, vol. 53, no. 5, pp. 1398–1409, 2006.

[69] M. Erol-Kantarci and H. T. Mouftah, "Wireless sensor networks for cost-efficient residential energy management in the smart grid," *IEEE Trans. on Smart Grid*, vol. 2, no. 2, pp. 314–325, 2011.

[70] C. R. Baker, K. Armijo, S. Belka, M. Benhabib, V. Bhargava, N. Burkhart, A. Der Minassians, G. Dervisoglu, L. Gutnik, M. B. Haick *et al.*, "Wireless sensor networks for home health care," in *Advanced Information Networking and Applications Workshops, 2007, AINAW'07. 21st Int. Conf. on*, vol. 2. IEEE, 2007, pp. 832–837.

[71] H. Alemdar and C. Ersoy, "Wireless sensor networks for healthcare: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2688–2710, 2010.

[72] F. Bénézit, V. Blondel, P. Thiran, J. Tsitsiklis, and M. Vetterli, "Weighted gossip: Distributed averaging using non-doubly stochastic matrices," in *Information theory proceedings (isit), 2010 IEEE Int. symposium on*. IEEE, 2010, pp. 1753–1757.

[73] D. Shuman, S. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Mag.*, vol. 30, no. 3, pp. 83–98, 2013.

[74] E. Isufi, A. Simonetto, A. Loukas, and G. Leus, "Stochastic graph filtering on time-varying graphs," in *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2015 IEEE 6th Int. Workshop on*. IEEE, 2015, pp. 89–92.

[75] Z. Luo and W. Yu, "An introduction to convex optimization for communications and signal processing," *IEEE Journal on selected areas in communications*, vol. 24, no. 8, pp. 1426–1438, 2006.

[76] R. Rockafellar, "Network flows and monotropic optimization," 1984.

[77] L. Condat, "A primal-dual splitting method for convex optimization involving lipschitzian, proximable and linear composite terms," *Journal of Opt. Theory and Applications*, vol. 158, no. 2, pp. 460–479, 2013.

[78] B. Vũ, "A splitting algorithm for dual monotone inclusions involving cocoercive operators," *Advances in Computational Mathematics*, vol. 38, no. 3, pp. 667–681, 2013.

[79] G. Scutari, F. Facchinei, and L. Lampariello, "Parallel and distributed methods for constrained nonconvex optimization-part i: Theory." *IEEE Trans. Signal Processing*, vol. 65, no. 8, pp. 1929–1944, 2017.

[80] G. Scutari, F. Facchinei, L. Lampariello, S. Sardellitti, and P. Song, "Parallel and distributed methods for constrained nonconvex optimization-part ii: Applications in communications and machine learning," *IEEE Trans. on Signal Processing*, vol. 65, no. 8, pp. 1945–1960, 2017.

[81] J. Eckstein and W. Yao, "Augmented Lagrangian and alternating direction methods for convex optimization: A tutorial and some illustrative computational results," *RUTCOR Research Reports*, vol. 32, p. 3, 2012.

[82] P. Giselsson and S. Boyd, "Linear convergence and metric selection for Douglas-Rachford splitting and ADMM," *IEEE Trans. on Automatic Control*, vol. 62, no. 2, pp. 532–544, 2017.

[83] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the ADMM in decentralized consensus optimization." *IEEE Trans. Signal Processing*, vol. 62, no. 7, pp. 1750–1761, 2014.

[84] N. Parikh, S. P. Boyd *et al.*, "Proximal algorithms." *Foundations and Trends in Opt.*, vol. 1, no. 3, pp. 127–239, 2014.

[85] P. Bianchi, W. Hachem, and F. Iutzeler, "A coordinate descent primal-dual algorithm and application to distributed asynchronous optimization," *ArXiv:1407:0898*, 2014.

[86] J. Eckstein and D. Bertsekas, "On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators," *Math. Programming*, vol. 55, no. 1-3, pp. 293–318, 1992.

[87] W. Deng and W. Yin, "On the global and linear convergence of the generalized alternating direction method of multipliers," *Journal of Scientific Computing*, vol. 66, no. 3, pp. 889–916, 2016.

[88] M. Fält and P. Giselsson, "Optimal convergence rates for generalized alternating projections," *arXiv preprint arXiv:1703.10547*, 2017.

[89] H. Bauschke, J. Cruz, T. Nghia, H. Pha, and X. Wang, "Optimal rates of linear convergence of relaxed alternating projections and generalized douglas-rachford methods for two subspaces," *Numerical Algorithms*, vol. 73, no. 1, pp. 33–76, 2016.

[90] P. Erdos and A. Rényi, "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci*, vol. 5, no. 1, pp. 17–60, 1960.

[91] M. Swan, *Blockchain: Blueprint for a new economy*. " O'Reilly Media, Inc.", 2015.

[92] F. Bénézit, A. Dimakis, P. Thiran, and M. Vetterli, "Gossip along the way: Order-optimal consensus through randomized path averaging," in *Allerton*, no. LCAV-CONF-2009-004, 2007.

[93] M. Mehyar, D. Spanos, J. Pongsajapan, S. Low, and R. Murray, "Asynchronous distributed averaging on communication networks," *IEEE/ACM Trans. on Networking (TON)*, vol. 15, no. 3, pp. 512–520, 2007.

[94] F. R. Kschischang, B. Frey, and H. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. on information theory*, vol. 47, no. 2, pp. 498–519, 2001.

[95] S. Chen, A. Sandryhaila, J. Moura, and J. Kovacevic, "Signal denoising on graphs via graph filtering," in *Signal and Information Processing (GlobalSIP), 2014 IEEE Global Conf. on*. IEEE, 2014, pp. 872–876.

[96] G. Zhang and R. Heusdens, "Distributed optimization using the primal-dual method of multipliers," *IEEE Trans. on Signal and Information Processing over Networks*, 2017.

[97] P. Latafat, L. Stella, and P. Patrinos, "New primal-dual proximal algorithm for distributed optimization," in *Decision and Control (CDC), 2016 IEEE 55th Conf. on*. IEEE, 2016, pp. 1959–1964.

[98] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[99] T. Sherson, R. Heusdens, and W. B. Kleijn, "Derivation and analysis of the primal-dual method of multipliers based on monotone operator theory," *arXiv preprint arXiv:1706.02654*, 2018.

[100] G. Barrenetxea, F. Ingelrest, G. Schaefer, and M. Vetterli, "Wireless sensor networks for environmental monitoring: The sensorscope experience," in *Communications, 2008 IEEE International Zurich Seminar on*. IEEE, 2008, pp. 98–101.

[101] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Transactions on Industrial informatics*, vol. 9, no. 1, pp. 427–438, 2013.

[102] A. Nedić, A. Olshevsky, and M. G. Rabbat, "Network topology and communication-computation tradeoffs in decentralized optimization," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 953–976, 2018.

[103] A. Agarwal, M. J. Wainwright, and J. C. Duchi, "Distributed dual averaging in networks," in *Advances in Neural Information Processing Systems*, 2010, pp. 550–558.

[104] F. Iutzeler, P. Bianchi, P. Ciblat, and W. Hachem, "Explicit convergence rate of a distributed alternating direction method of multipliers," *IEEE Transactions on Automatic Control*, vol. 61, no. 4, pp. 892–904, 2016.

[105] G. França and J. Bento, "Markov chain lifting and distributed ADMM," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 294–298, 2017.

[106] ——, "How is distributed admm affected by network topology?" *arXiv preprint arXiv:1710.00889*, 2017.

[107] C. Jordan, "Essai sur la géométriean dimensions," *Bull. Soc. Math. France*, vol. 3, pp. 103–174, 1875.

[108] R. A. Horn, R. A. Horn, and C. R. Johnson, *Matrix analysis*. Cambridge university press, 1990.

[109] F. R. Chung, *Spectral graph theory*. American Mathematical Soc., 1997, no. 92.

[110] S. Barik, R. B. Bapat, and S. Pati, "On the Laplacian spectra of product graphs," *Applicable Analysis and Discrete Mathematics*, pp. 39–58, 2015.

[111] G. Lu and W. H. Zeng, "Cloud computing survey," in *Applied Mechanics and Materials*, vol. 530. Trans. Tech. Publ, 2014, pp. 650–661.

[112] R. Bekkerman, M. Bilenko, and J. Langford, *Scaling up machine learning: Parallel and distributed approaches*. Cambridge University Press, 2011.

[113] D. Davis and W. Yin, "A three-operator splitting scheme and its optimization applications," *Set-valued and variational analysis*, vol. 25, no. 4, pp. 829–858, 2017.

[114] P. L. Combettes and V. R. Wajs, "Signal recovery by proximal forward-backward splitting," *Multiscale Modeling & Simulation*, vol. 4, no. 4, pp. 1168–1200, 2005.

[115] V. Berinde, *Iterative approximation of fixed points*. Springer, 2007, vol. 1912.

[116] R. Rockafellar, "On the maximality of sums of nonlinear monotone operators," *Trans. of the American Mathematical Society*, vol. 149, no. 1, pp. 75–88, 1970.

[117] P. Erdös and A. Rényi, "On random graphs, i," *Publicationes Mathematicae (Debrecen)*, vol. 6, pp. 290–297, 1959.

[118] D. J. Watts and S. H. Strogatz, "Collective dynamics of "Small-World" networks," *nature*, vol. 393, no. 6684, p. 440, 1998.

[119] M. Penrose, *Random geometric graphs*. Oxford university press, 2003, no. 5.

[120] P. Gupta and P. Kumar, "The capacity of wireless networks," *IEEE Trans. on Information Theory*, vol. 46, no. 2, pp. 388–404, 2000.

[121] H. L. Van Trees, *Optimum array processing: Part IV of detection, estimation, and modulation theory*. John Wiley & Sons, 2004.

[122] C. Shannon, "Communication in the presence of noise," *Proceedings of the IRE*, vol. 37, no. 1, pp. 10–21, 1949.

[123] H. Markowitz, "Portfolio selection," *The journal of finance*, vol. 7, no. 1, pp. 77–91, 1952.

[124] A. Nedic, A. Olshevsky, and W. Shi, "Achieving geometric convergence for distributed optimization over time-varying graphs," *SIAM Journal on Optimization*, vol. 27, no. 4, pp. 2597–2633, 2017.

[125] A. Nedic, A. Olshevsky, A. Ozdaglar, and J. N. Tsitsiklis, "Distributed subgradient methods and quantization effects," in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*. IEEE, 2008, pp. 4177–4184.

[126] B. D. Van Veen and K. M. Buckley, "Beamforming: A versatile approach to spatial filtering," *IEEE ASSP Mag.*, vol. 5, no. 5, pp. 4–24, Apr. 1988.

[127] H. L. Van Trees, *Detection, Estimation, and Modulation Theory, Optimum Array Processing*. John Wiley & Sons, 2004.

[128] S. A. Vorobyov, "Principles of minimum variance robust adaptive beamforming design," *ELSEVIER Signal Process.*, vol. 93, no. 12, pp. 3264–3277, Dec. 2013.

[129] J. Benesty, M. M. Sondhi, and Y. Huang (Eds), *Springer handbook of speech processing*. Springer, 2008.

[130] M. Brandstein and D. Ward (Eds.), *Microphone arrays: signal processing techniques and applications*. Springer, 2001.

[131] P. Vary and R. Martin, *Digital speech transmission: Enhancement, coding and error concealment*. John Wiley & Sons, 2006.

[132] S. Gannot, E. Vincet, S. Markovich-Golan, and A. Ozerov, "A consolidated perspective on multi-microphone speech enhancement and source separation," *IEEE Trans. Audio, Speech, Language Process.*, vol. 25, no. 4, pp. 692–730, April 2017.

[133] A. Bertrand, "Applications and trends in wireless acoustic sensor networks: A signal processing perspective," in *18th IEEE Symp. on Comm. and Vehicular Tech.*, Nov. 2011, pp. 1–6.

[134] Y. Zeng and R. C. Hendriks, "Distributed delay and sum beamformer for speech enhancement via randomized gossip," *IEEE Trans. Audio, Speech, Language Process.*, vol. 22, no. 1, pp. 260–273, Jan. 2014.

[135] R. Heusdens, G. Zhang, R. C. Hendriks, Y. Zeng, and W. B. Kleijn, "Distributed MVDR beamforming for (wireless) microphone networks using message passing," in *Int. Workshop Acoustic Signal Enhancement (IWAENC)*, Sep. 2012, pp. 1–4.

[136] M. O'Connor and W. B. Kleijn, "Diffusion-based distributed MVDR beamformer," in *IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, May 2014, pp. 810–814.

[137] M. O'Connor, W. B. Kleijn, and T. Abhayapala, "Distributed sparse MVDR beamforming using the bi-alternating direction method of multipliers," in *IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Mar. 2016.

[138] A. Bertrand and M. Moonen, "Distributed node-specific LCMV beamforming in wireless sensor networks," *IEEE Trans. Signal Process.*, vol. 60, no. 1, pp. 233–246, Sep. 2012.

[139] ——, "Distributed LCMV beamforming in a wireless sensor network with single-channel per-node signal transmission," *IEEE Trans. Signal Process.*, vol. 61, no. 13, pp. 3447–3459, Apr. 2013.

[140] S. Markovich, S. Gannot, and I. Cohen, "Distributed multiple constraints generalized sidelobe canceler for fully connected wireless acoustic sensor networks," *IEEE Trans. Audio, Speech, Language Process.*, vol. 21, no. 2, pp. 343–356, Oct. 2013.

[141] T. Sherson, W. B. Kleijn, and R. Heusdens, "A distributed algorithm for robust LCMV beamforming," in *IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Mar. 2016.

[142] S. Doclo, M. Moonen, T. V. den Bogaert, and J. Wouters, "Reduced-bandwidth and distributed MWF-based noise reduction algorithms for binaural hearing aids," *IEEE Trans. Audio, Speech, Language Process.*, vol. 17, no. 1, pp. 38–51, Jan. 2009.

[143] J. Szurley, A. Bertrand, and M. Moonen, "Topology-independent distributed adaptive node-specific signal estimation in wireless sensor networks," *IEEE Trans. Signal and Info. Process. Over Networks*, vol. 3, no. 1, pp. 130–144, 2017.

[144] J. Capon, "High-resolution frequency-wavenumber spectrum analysis," *Proc. of the IEEE*, vol. 57, no. 8, pp. 1408–1418, Aug. 1969.

[145] O. L. Frost III, "An algorithm for linearly constrained adaptive array processing," *Proc. of the IEEE*, vol. 60, no. 8, pp. 926–935, Aug. 1972.

[146] H. Cox, "Resolving power and sensitivity to mismatch of optimum array processors," *J. Acoust. Soc. Amer.*, vol. 54, no. 3, pp. 771–785, Sep. 1973.

[147] R. C. Hendriks and T. Gerkmann, "Noise correlation matrix estimation for multi-microphone speech enhancement," *IEEE Trans. Audio, Speech, Language Process.*, vol. 20, no. 1, pp. 223–233, Jan. 2012.

[148] H. Cox, "Robust adaptive beamforming," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-35, no. 10, pp. 1365–1376, Oct. 1987.

[149] B. D. Carlson, "Covariance matrix estimation errors and diagonal loading in adaptive arrays," *IEEE Trans. Aerosp. Electron. Systems*, vol. 24, no. 4, pp. 397–401, July 1988.

[150] J. Li, P. Stoica, and Z. Wang, "On robust Capon beamforming and diagonal loading," *IEEE Trans. Signal Process.*, vol. 51, no. 7, pp. 1702–1715, July 2003.

[151] V. M. Tavakoli, J. R. Jensen, R. Heusdens, J. Benesty, and M. G. Christensen, "Ad hoc microphone array beamforming using the primal-dual method of multipliers," in *EURASIP Europ. Signal Process. Conf. (EUSIPCO)*. IEEE, 2016, pp. 1088–1092.

[152] J. L. Flanagan, A. C. Surendran, and E. E. Jan, "Spatially selective sound capture for speech and audio processing," *ELSEVIER Speech Commun.*, vol. 13, no. 1-2, pp. 207–222, Oct. 1993.

[153] S. Gannot, D. Burshtein, and E. Weinstein, "Signal enhancement using beamforming and nonstationarity with applications to speech," *IEEE Trans. Signal Process.*, pp. 1614–1626, Aug. 2001.

[154] J. S. Bradley, "Predictors of speech intelligibility in rooms," *J. Acoust. Soc. Amer.*, vol. 80, no. 3, pp. 837–845, Sept. 1986.

[155] J. S. Bradley and H. Sato, "On the importance of early reflections for speech in rooms," *J. Acoust. Soc. Amer.*, vol. 113, no. 6, pp. 3233–3244, June 2003.

[156] S. Gannot and I. Cohen, "Speech enhancement based on the general transfer function GSC and postfiltering," *IEEE Trans. Speech Audio Process.*, pp. 561–571, Nov. 2004.

[157] S. Markovich, S. Gannot, and I. Cohen, "Multichannel eigenspace beamforming in a reverberant noisy environment with multiple interfering speech signals," *IEEE Trans. Audio, Speech, Language Process.*, pp. 1071–1086, Aug. 2009.

[158] A. Bertrand and M. Moonen, "Distributed adaptive generalized eigenvector estimation of a sensor signal covariance matrix pair in a fully connected sensor network," *ELSEVIER Signal Process.*, vol. 106, pp. 209–214, Jan. 2015.

[159] S. Braun and E. A. P. Habets, "Dereverberation in noisy environments using reference signals and a maximum likelihood estimator," in *EURASIP Europ. Signal Process. Conf. (EUSIPCO)*, Sep. 2013.

[160] M. Souden, J. Benesty, and S. Affes, "A study of the LCMV and MVDR noise reduction filters," *IEEE Trans. Signal Process.*, vol. 58, no. 9, pp. 4925–4935, Sep. 2010.

[161] E. Hadad, S. Doclo, and S. Gannot, "The binaural LCMV beamformer and its performance analysis," *IEEE Trans. Audio, Speech, Language Process.*, vol. 24, no. 3, pp. 543–558, Jan. 2016.

[162] J. Bitzer, K. U. Simmer, and K. Kammeyer, "Theoretical noise reduction limits of the generalized sidelobe canceler (GSC) for speech enhancement," in *IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, vol. 5, March 1999, pp. 2965–2968.

[163] I. A. McCowan and H. Bourlard, "Microphone array post-filter based on noise field coherence," *IEEE Trans. Audio, Speech, Language Process.*, vol. 11, no. 6, pp. 709–716, Nov. 2003.

[164] E. N. Gilbert and S. P. Morgan, "Optimum design of directive antenna arrays subject to random variations," *Bell Labs Technical Journal*, vol. 34, no. 3, pp. 637–663, May 1955.

[165] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the world with wireless sensor networks," in *IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, vol. 4, May 2001, pp. 2033–2036.

[166] D. Brandwood, "A complex gradient operator and its application in adaptive array theory," *IEE Proc. Pts. F and H*, vol. 130, no. 1, pp. 11–16, Feb. 1983.

[167] S. Markovich, A. Bertrand, M. Moonen, and S. Gannot, "Optimal distributed minimum-variance beamforming approaches for speech enhancement in wireless acoustic sensor networks," *ELSEVIER Signal Process.*, vol. 107, pp. 4–20, Feb. 2015.

[168] J. B. Allen and D. A. Berkley, "Image method for efficiently simulating small-room acoustics," *J. Acoust. Soc. Amer.*, vol. 65, no. 4, pp. 943–950, Apr. 1979.

[169] E. A. P. Habets, "Room impulse response generator," https://www.audiolabs-erlangen.de/fau/professor/habets/software/rir-generator/, 2010.

[170] J. J. Shynk, "Frequency-domain and multirate adaptive filtering," *IEEE Signal Process. Mag.*, vol. 9, no. 1, pp. 14–37, Jan. 1992.

[171] T. Drugman, Y. Stylianou, Y. Kida, and M. Akamine, "Voice activity detection: Merging source and filter-based information," *IEEE Signal Process. Lett.*, vol. 23, no. 2, pp. 252–256, 2016.

[172] C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen, "An algorithm for intelligibility prediction of time-frequency weighted noisy speech," *IEEE Trans. Audio, Speech, Language Process.*, vol. 19, no. 7, pp. 2125–2136, Sep. 2011.

[173] C. Kanzow and Y. Shehu, "Generalized krasnoselskii–mann-type iterations for nonexpansive mappings in hilbert spaces," *Computational Optimization and Applications*, vol. 67, no. 3, pp. 595–620, 2017.

[174] J. A. G. Jonkman, T. Sherson, and R. Heusdens, "Quantisation effects in distributed optimisation," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, April 15-20, 2018*, 2018, pp. 3649–3653. [Online]. Available: https://doi.org/10.1109/ICASSP.2018.8461782

CURRICULUM VITÆ

Thomas William Sherson was born in Petersfield, United Kingdom on the 30th of March 1992. His high school education was undertaken at Mount Aspiring College, Wanaka, New Zealand from 2004 to 2010. In 2011 he moved to Wellington, New Zealand where he undertook his bachelor of engineering degree at Victoria University of Wellington which he completed with first-class Honors, majoring in electrical and computer systems engineering, in 2014. During this time, he took part in an exchange to Limerick University, Limerick, Ireland as part of the Industrialized Countries Instrument Education Cooperation Program (ICIEP) in 2012 where he worked as a researcher developing low cost sensor nodes for remote animal tracking. In 2013 he worked as a research assistance for Victoria University of Wellington while during his Honors year (2014) he worked with Callaghan Innovation, Wellington, New Zealand, on the development of high-voltage, low-current power supplies for medical microfluidic applications. He was also the recipient of the 2015 Victoria University Medal of Academic Excellence for his work during his undergraduate studies.

In January of 2015, Thomas joined the Circuits and Systems (CAS) group at Delft University of Technology, Delft, The Netherlands to pursue his PhD under the supervision of Professor W. Bastiaan Kleijn and Professor Richard Heusdens. This research was part of the "Distributed Processing of Audio Signals" project sponsored by Huawei and focused on the area of distributed signal processing and in particular distributed convex optimization. In 2018, he visited the Electrical and Computer Engineering department at the University of California, Los Angeles (UCLA) under the supervision of Professor Lieven Vandenberghe. His research interests include the likes of signal processing in wireless sensor networks, distributed/decentralized optimization, monotone operator theory, and audio signal processing. Additionally, he is an avid outdoorsman with a passion for nature, particularly mountains, and a love for music and music technology.