On the Primal-Dual Method of Multipliers and its Applications

by

Matthew O'Connor

A thesis
submitted to the Victoria University of Wellington
and the Australian National University
in fulfilment of the
requirements for the degree of
Doctor of Philosophy
in Engineering.

Victoria University of Wellington &
The Australian National University
2018

Abstract

With ever growing sources of digital data and the reductions in cost of small-scale wireless processing nodes, equipped with various sensors, microprocessors, and communication systems, we are seeing an increasing need for efficient distributed processing algorithms and techniques. This thesis focuses on the Primal-Dual Method of Multipliers (PDMM) as it applies to wireless sensor networks, and develops new algorithms based on PDMM more appropriate for the limitations on processing power, battery life, and memory that these devices suffer from. We develop FS-PDMM and QA-PDMM that greatly improve the efficiency of local node computations when dealing with regularized optimization problems and smooth cost function optimization problems, respectively. We combine these approaches to form the FSQA-PDMM algorithm that may be applied to problems with smooth cost functions and non-smooth regularization functions. Additionally, these three methods often eliminate the need for numerical optimization packages, reducing the memory cost on our nodes. We present the FT-PDMM algorithm for finite-time convergence of quadratic consensus problems, reducing the number of in-network iterations required for network convergence. Finally, we present two signal processing applications that benefit from our theoretical work: a distributed sparse near-field acoustic beamformer; and a distributed image fusion algorithm for use in imaging arrays. Simulated experiments confirm the benefit of our approaches, and demonstrate the computational gains to be made by tailoring our techniques towards sensor networks.

Acknowledgments

I would like to thank Professor Bastiaan Kleijn and Professor Thushara Abhayapala for the guidance and support they have provided me over the duration of my postgraduate studies, it has been invaluable. I would additionally like to thank Victoria University of Wellington and the Australian National University for affording me the opportunity of undertaking a dual PhD between the institutes. I have formed networks and met peers I would never have otherwise. Finally, I would like to thank Rebekah for supporting me through my studies and in particular my final year. I couldn't have done it without you.

Publications of the thesis

The contents of this thesis have been published or will appear in the form of the following papers,

I M. O'Connor, W. B. Kleijn, T. Abhayapala, "Distributed Sparse MVDR Beamforming using the Bi-Alternating Direction Method of Multipliers", in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* (March 2016)

II M. O'Connor, W. B. Kleijn, T. Abhayapala, "Distributed TV-L1 Image Fusion using PDMM", in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* (March 2017)

III M. O'Connor, G. Zhang, W. B. Kleijn, T. Abhayapala, "Function Splitting and Quadratic Approximation of the Primal-Dual Method of Multipliers for Distributed Optimization over Graphs", in *IEEE Transactions on Signal and Information Processing over Networks (TSIPN).* (December 2018)

IV G. Zhang, M. O'Connor, "Finite-Time Convergence of Asynchronous Primal-Dual Method of Multipliers for Quadratic Consensus Optimization", *Submitted for publication.*

V G. Zhang, M. O'Connor, L. Li, "On Convergence Analysis of Gradient Based Primal-Dual Method of Multipliers", in *IEEE Statistical Signal Processing Workshop (SSP).* (June 2018)

VI M. O'Connor, G. Zhang, W. B. Kleijn, T. Abhayapala, "The FSQA-PDMM Algorithm for Regularized Optimization over Networks using Inexact Updates", *To be submitted for publication.*

List of Common Notation

$\mathcal{V}$	set of all nodes, or vertices, in our network
$\mathcal{E}$	set of all directionless edges between node pairs
$\mathcal{V}_i$	set of all nodes neighbouring node i
$\boldsymbol{x}_i^k$	local primal optimization vector held by node i at iteration k
$\boldsymbol{\lambda}_{i\|j}^k$	local dual optimization vector held by node i relating to neighbour j at iteration k
$(\boldsymbol{x}_i^*, \boldsymbol{\lambda}_{i\|j}^*)$	optimal primal and dual point for node i related to neighbour j
$\boldsymbol{A}_{i\|j}$	constraint matrix held by node i relating to neighbour j
$\boldsymbol{P}_{ij}$	tuning parameter matrix for the edge (i, j), notated ρ if it is a common scalar over all edges
$f_i(\boldsymbol{x}_i)$	cost function of node i with local primal variable $\boldsymbol{x}_i$
$\mathbf{E}[Y]$	expected value of random variable Y
$\|\boldsymbol{y}\|_p$	the p-norm of vector $\boldsymbol{y}$
$\|\boldsymbol{y}\|_{\boldsymbol{M}}^2$	the weighted vector norm $\boldsymbol{y}^T \boldsymbol{M} \boldsymbol{y}$

Contents

Chapter 1

Introduction

1.1 Motivation

Imagine the following: a distant planet, yet to feel the footstep of its first human visitor, is orbited by a spacecraft. Before sending a brave soul down for exploration, it would seem prudent to perform a preliminary investigation of the planet's surface. How may we accomplish this? One approach would be to send a sophisticated robot down to roam around and report back to the satellite. However, this may be costly and for basic measurements of the planets magnetic field or surface temperature it may be unecessary. Another option is to fire out hundreds of small, inexpensive, wireless processing units equipped with microprocessors, transceivers for communication, and thermal, acoustic, and pressure sensors for measurement of the planet. These self contained units would be capable of collaboratively and distributively collecting data from the planet. Return transmission of this data could then be accomplished by collectively transmitting the measurements to the orbiting mothership.

Science fiction? Not necessarily. We are, in fact, witnessing the birth of these exact systems. An example more closely realisable may be distributed public recording of sound, allowing large wireless microphone networks to isolate sources of interest and reduce background interfer-

ence. Networks such as these could occur, for example, when a large crowd at a music concert wishes to collaboratively record a performance using their smart phones.. Over the last few years electronic sensor technology has seen a rapid increase in processing power as well as a decrease in component cost. This coupled with recent advances in wireless communication technology has made feasible the development of large scale wireless sensor networks for use in speech detection, interference cancellation and temperature sensing, among others. Ad-hoc networks, with random topologies, are particularly attractive as mobile sensor devices (e.g., smart phones) are ubiquitous and self-localisation is now becoming possible (e.g., [45]). These developments in hardware and software are finally allowing for practical research into distributed systems, with the final piece of the puzzle being the distributed algorithms necessary for performing useful tasks over geographically separated networks.

In recent years there has been an increasing focus on distributed optimization methods, driven by areas such as cloud computing [143], big data [21, 109], and wireless sensor networks [107]. The growing size of current databases and sensor networks are reducing the effectiveness of centralized processing schemes, due to the large volume of data as well as the geographically distributed sensors often present in these systems. Many of these approaches exploit dual decomposition techniques [102, 16] to distribute computations among processing units, that use gradient methods [67, 148] for local objective function minimization. Incremental local optimization and parameter sharing throughout the network leads to the minimization of global utility.

Existing methods in the literature for distributed processing can be roughly classified into three categories: distributed averaging methods, belief propagation methods, and methods developed for decomposable convex and nonconvex optimization problems. Decomposable problems, in particular, have seen increasing attention in the past decade as a flexible and efficient framework for distributed processing [15, 157, 61, 32, 96, 26].

Two closely related methods for solving decomposable optimization problems are the Primal-Dual Method of Multipliers (PDMM) [158, 153] and the popular Alternating Direction Method of Multipliers (ADMM) [15, 41, 48]. These both exhibit a practically useful tradeoff between local computational complexity and global network convergence speed when solving distributed problems, leading to their appropriateness for distributed processing. These approaches have found use in distributed acoustic processing [98, 130], network power flow optimization [34], communications [111, 71, 160, 119], and distributed machine learning [15], among others. However, distributed wireless sensor networks are naturally very limited when it comes to local computation power, battery life, and available optimization packages due to memory restrictions. This increases the need for local computational simplicity when designing algorithms for these systems.

1.2 Contributions of the thesis

In this thesis we consider improving on the applicability of PDMM for distributed optimization over wireless sensor networks. This improvement will come in three forms: a reduction in the necessary local computation required by improving algorithmic efficiency; an elimination in the need for optimization packages to be held at individual sensor nodes for a common class of problems; and a reduction in total network processing iterations required for accurate problem solutions. Additionally, we develop practical signal processing algorithms in the areas of acoustics and imaging that utilize some of our methods in order to demonstrate their real world use.

We propose three variations on PDMM, and experimentally analyse a fourth, each with different application strengths. Firstly, we develop the asynchronous Function Splitting PDMM (FS-PDMM) algorithm for use in regularized optimization problems. We show that local computational

efficiency is significantly improved for coupled optimization problems, while also eliminating the need for numerical solution packages. Secondly, we develop and analyse the synchronous Quadratic Approximation (QA-PDMM) algorithm for use in optimization problems where local cost functions are Lipschitz smooth. We show that for smooth problems this algorithm also improves local computational efficiency while eliminating the need for optimization packages. Thirdly, we combine the FS-PDMM and QA-PDMM algorithms in order to expand the applicability of the methods to a larger class of optimization problems. We show that the synchronous FSQA-PDMM algorithm benefits from both aspects of the constituent algorithms, leading to greater improvements when applied to problems with a smooth cost function and a non-smooth regularization function. We experimentally study the final variation, Finite-Time PDMM (FT-PDMM), for use in distributed quadratic problems. For quadratic consensus problems, we show that FT-PDMM converges in finite time rather than sub-linearly reducing optimization variable error. Finally, we present two signal processing applications for these algorithms. We develop a distributed sparse near-field acoustic beamformer and a distributed image fusion approach, for use in wireless sensor networks, and demonstrate how these benefit from our theoretical developments.

We summarize the contributions as follows

- The development and analysis of FS-PDMM for efficient distributed regularized optimization
- The development and analysis of QA-PDMM for efficient distributed optimization with smooth functions
- The combination and analysis of the above two algorithms, resulting in FSQA-PDMM for optimization of functions with a smooth cost function and non-smooth regularization function.
- The experimental analysis of FT-PDMM for optimization of distributed quadratic problems in finite iterations.

- The development of a sparse near-field acoustic beamformer that uses regularization to naturally select an optimal subset of network nodes for beamformer output.

- The development of a distributed image fusion algorithm that allows geographically distant imaging sensors with partially overlapping fields of view to fuse common imaging regions, thereby improving image fidelity.

1.3 Structure of the thesis

The remainder of this thesis is organized as follows. Chapter 2 provides notation, the wireless sensor network system model, and background on distributed processing, beamforming, and distributed beamforming. Chapter 3 presents the problem formulation and simulation of a preliminary distributed PDMM beamformer, and motivates the need for more efficient distributed algorithms.

Chapter 4 (included in publication III) presents the FS-PDMM algorithm, including application to the problem of distributed elastic net regularized least-squares and an equivalence analysis. Chapter 5 (included in publication III & V) motivates and develops the QA-PDMM algorithm, which is analysed and then applied to distributed logistic regression. Chapter 6 (included in publication VI) combines the two previous algorithms into the FSQA-PDMM algorithm, analysing convergence and applying it to the problem of sparse logistic regression. Chapter 7 (included in publication IV) summarizes the FT-PDMM algorithm and applies it to the distributed quadratic consensus problem, studying its experimental convergence.

Chapter 8 (included in publications I & II) presents a distributed sparse beamformer and a distributed image fusion algorithm, employing some of the methods discussed in earlier chapters. Finally, we summarized the

thesis in chapter 9, including the possibility of future work.

Chapter 2

Background

In this chapter we cover the general mathematical notation used throughout the thesis and describe how the distributed sensor network is modelled. We then provide an overview of distributed processing techniques and how these pertain to solving problems over sensor networks. We then provide a background on beamforming, a signal processing method that will serve as motivation for algorithms developed later in the thesis.

2.1 Summary of Notation and Wireless Sensor Networks

We begin by presenting the notation used in the remainder of this thesis and by establishing the mathematical model used to describe our wireless sensor networks (WSN). We will then briefly summarize the current state of WSN technology that allows for the design and implementation of distributed signal processing systems. This section aims to set the stage for the distributed processing techniques that appear later in the chapter.

2.1.1 Summary of Notation

The notation throughout this thesis will abide by this section, except where otherwise stated. Vectors will be denoted by lowercase bold italics, matrices will be denoted by uppercase bold italics, while scalars will use regular font. Blackboard bold font, such as $\mathbb{R}$ will be used to refer to the dimensionality of our vectors. Therefore $\boldsymbol{w} \in \mathbb{R}^V$ denotes a vector of length V, $\boldsymbol{A} \in \mathbb{R}^{V \times V}$ denotes a $V \times V$ matrix, and $\alpha \in \mathbb{R}$ denotes a real scalar. All vectors are considered to be column vectors. Calligraphic letters will refer to sets and the scalar cardinality of a set will be uppercase normal font. In this way, the set $\mathcal{V}$ has scalar cardinality V. In the context of optimization, Greek letters will generally refer to dual variables, or Lagrange multipliers. Therefore a problem with primal vector variable $\boldsymbol{x}$ may have the dual vector variable $\boldsymbol{\lambda}$. Subscripts will be used to denote ownership by a sensor node, such as node i's weight vector $\boldsymbol{x}_i$. Superscripts will refer to update indices in iterative algorithms and parentheses will refer to time indices. In this way, $\boldsymbol{w}_i^k$ is node i's weight vector at update iteration k while $u_k(t)$ is the scalar observation of node k at time sample t.$(\cdot)^T$ will refer to vector or matrix transposition and $(\cdot)^*$ will refer to optimal points when considering optimization procedures. We will refer to component l of vector $\boldsymbol{x}_i$ as $[\boldsymbol{x}_i]_l$, assign ∇ the role of the derivative operator, and denote the expected value of a random variable $\mathbf{E}[\cdot]$. The regular font letters f, g, R, and L refer to functions, where L denotes the Lagrangian function. When referring to the convergence rate of an algorithm we will use big O notation [124] such as $O(1/k)$ to refer to an algorithm with a convergence rate that improves linearly in iterations k. $\boldsymbol{I}$ denotes an identity matrix. The notation $\boldsymbol{M} \succeq 0$ (or $\boldsymbol{M} \succ 0$) represents a symmetric positive semi-definite matrix (or a symmetric positive definite matrix) while $(\cdot)^{-1}$ represents matrix inversion. Given a vector $\boldsymbol{y}$, we use $\|\boldsymbol{y}\|$ to denote its l_2 norm. Furthermore, for a positive-definite matrix M, where $M = P^T P$, the weighted l_2 norm of a vector $\boldsymbol{y}$ is notated as $\|\boldsymbol{y}\|_M = \|P\boldsymbol{y}\|_2$. Consequently, we have that the square of this weighted norm gives $\|\boldsymbol{y}\|_M^2 = \boldsymbol{y}^T \boldsymbol{M} \boldsymbol{y}$.

2.1.2 Modelling the Physical Network

We consider a network of nodes, or vertices, denoted by the set $\mathcal{V}$ with cardinality $V = |\mathcal{V}|$. The network is connected by a set of edges $\mathcal{E}$ with cardinality $E = |\mathcal{E}|$. If there exists an edge between two nodes i and j form the unordered pair $\{(i, j\} \in \mathcal{E}$. The node and edge sets together form our network graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The set of all nodes sharing an edge with a node i is denoted as its neighbourhood $\mathcal{V}_i$ with cardinality $V_i = |\mathcal{V}_i|$, i.e. $\mathcal{V}_i = \{j|\{i, j\} \in \mathcal{E}\}$. Each node i carries a vector of length n_i denoted as $\boldsymbol{x}_i \in \mathbb{R}^{n_i}$. As an example, consider figure 2.1 with $V = 9$ randomly connected nodes. The neighbourhood of node 8 is the set $\{8, 2, 3, 6\}$. This implies that node

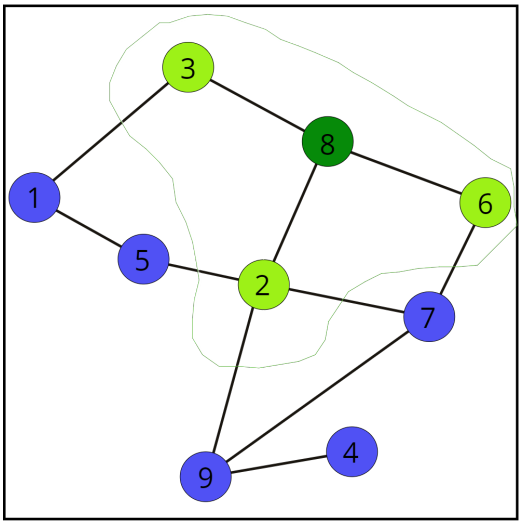


Figure 2.1: Network of nodes showing the local neighbourhood of node 8.

8 is able to communicate with nodes $\{8, 2, 3, 6\}$ in a single communication step.

2.1.3 Distributed processing building blocks

For any form of spatial processing to be applied we first require two forms of localization: self localization and source localization. Self localization refers to the ability of every node to determine its position with respect to

its neighbours while source localization refers to estimating the position of a source of interest, in our case a sound source, relative to each node. In recent years there has been substantial development on distributed self localization [37, 72, 30, 68, 45] and source localization [106, 84] algorithms in sensor networks. Nodes therefore know which other nodes are within their communication neighbourhood and where they should be listening for a source signal.

The final building block for considering distributed sensor network audio processing is a method of synchronizing sensor clocks. It is often necessary that signal samples are recorded synchronously to ensure the effectiveness of signal processing methods, such as beamforming. This is an important fundamental challenge in distributed systems a distant nodes in large networks may be out of synchronization on the order of whole time samples with sensor clocks drifting randomly. Approaches such as [115] and [81] make use of message exchange and gossip algorithms to estimate neighbouring node clock differences and correct these differences based on a virtual master clock. These approaches improve clock synchronicity, particularly for nodes near to one another, but as nodes become more distant this relative difference becomes more pronounced.

Some signal processing methods additionally require the estimation of network covariances. There has been some progress in distributed covariance estimation techniques [142], though these methods often make the simplifying assumption of covariance only between connected nodes. This allows each neighbourhood to function as a fully connected set of sensors for the purpose of centrally estimating neighbourhood covariance matrices, or, as in [142], these neighbourhood covariances may be estimated in a distributed manner using belief propagation (discussed in section 2.2.2). Once each node has an estimate of the covariance matrix for its neighbourhood the collection of these matrices may be used as a distributed sparse approximation of the true centralized covariance matrix, as was used in [97]. This approximation is poor, however, when the sen-

sor communication range is low relative to the audible range of the audio source signal since acoustic interference correlation may extend beyond the immediate neighbourhood. Further research in this area may lead to better approximations for this distributed covariance estimation.

2.1.4 Implications for Distributed Signal Processing

The results of the areas of research in this section, along with hardware advances in microprocessor and sensor technology, provide the necessary building blocks for a distributed signal processing system. Nodes know where other nodes are; they know where the source is located; they can estimate local noise covariances; they are operating on signal samples taken at roughly the same time; and they are equipped with sensors, microprocessors, and low range electromagnetic transmitters and receivers. We may now move on to distributed processing techniques that solve in-network problems.

2.2 Distributed Processing

The wireless sensor networks discussed in section 2.1 provide the necessary infrastructure for advanced signal processing techniques such as distributed target tracking, distributed blind source separation, and distributed beamforming which may often be framed as specific cases of more general distributed optimization problems. We will now introduce the most popular and recent approaches to solving these optimization problems as well as discuss their performance with regards to convergence rates, synchronicity, communication requirements, computational complexity, simplicity, and ability to deal with optimization constraints. Existing methods in the literature for distributed processing can be roughly classified into three categories: distributed averaging methods, belief propagation methods, and methods developed for decomposable convex and

nonconvex optimizations. Decomposable convex and nonconvex optimizations, in particular, have seen increasing attention in the past decade as a flexible and efficient framework for distributed signal processing [15, 158, 61, 32, 96, 26]. The basic idea is to rephrase a signal processing problem as the minimization of a sum of individual objective functions subject to local constraints between neighbouring nodes. Distributed averaging, for example, can be formulated as a quadratic consensus optimization [158] problem.

First, within the category of distributed averaging, we will cover randomized gossip algorithms. We will then give an overview of general belief propagation, followed by three methods for decomposable optimization: distributed subgradient methods; the alternating direction method of multipliers (ADMM); and the primal-dual method of multipliers (PDMM).

2.2.1 Distributed Averaging using Randomized Gossip Algorithms (RGA)

Distributed averaging methods [14, 86, 31] intend to perform averaging across a graph distributedly and iteratively using only local communication between neighbours. After algorithmic convergence, every node in the graph has either an approximate or the exact averaging value. The simplest of these algorithms is a randomized pairwise protocol to perform distributed averaging [14, 13, 31]. Given a randomly connected network of V nodes with initial scalar value $g_i(0)$ at node i the randomized gossip algorithm (RGA) aims to find the average of these values $g_{\text{ave}} = \frac{1}{V}\sum_{k=1}^{V} g_i(0)$ through iterative pairwise communication. Asymptotically, all nodes will approach the average value of the network [14].

Let $g^k = [g_1^k, \cdots, g_V^k]^T$ denote the scalar values of all V nodes at the end of iteration k, where $(\cdot)^T$ denotes vector transposition. All nodes are equipped with Poisson clocks[1] that run independently at each iteration.

[1] A Poisson clock is an exponentially distributed timer with a mean value of λ seconds.

When node i's clock ticks it randomly selects one of its neighbours, j, with probability $p_{i,j}$. These probabilities are stored in an $V \times V$ right stochastic matrix $\boldsymbol{C}$ with nonzero entries where a connection exists between node i and node j, and zero entries where there are no links. Nodes i and j then set their values to the average of their previous values, i.e., $g_i^{k+1} = g_j^{k+1} = (g_i^k + g_j^k)/2$. This can be expressed in vector notation as

$$g^{k+1} = \boldsymbol{U}^k g^k \tag{2.1}$$

where $\boldsymbol{U}^k$ is an $V \times V$ update matrix selected independently across time, given by

$$\boldsymbol{U}^k = I_V - \frac{1}{2}(e_i - e_j)(e_i - e_j)^T, \tag{2.2}$$

where $e_i = [0, \cdots, 0, 1, 0, \cdots, 0]^T$ is an V dimensional vector with element i equal to 1 and $\boldsymbol{I}_V$ is the $V \times V$ identity matrix. Since the total 'mass' of the values across the network are conserved at each iteration, this pairwise averaging drives the nodes to converge on the global average with convergence time proportional to the second largest eigenvalue of the expected value of the update matrix $\mathbf{E}[\boldsymbol{U}]$ [14]. Convergence is therefore dependent on the connectivity of the network, with less connected networks converging slower. For the best case of a fully connected network the averaging time increases linearly as a function of the number of nodes V, or $O(V)$ [31] in 'big O' notation, while for the worst case of a line connected graph the averaging time increases substantially to $O(V^3)$ [149].

In recent years, a number of algorithms have been proposed which are able to perform exact averaging in finite iterations by smartly making use the properties of the graphic topology (see [24, 112, 113]), however these generally require an additional overhead prior to processing. Applications of distributed averaging include distributed speech enhancement [150], distributed detection [73], and distributed source localization [31]. RGA is a simple and computationally efficient method to implement and it allows for asynchronous iterations. However, the simplicity comes with limitations since we only perform a very basic distributed operation (averaging)

at a mediocre convergence rate. The algorithm may be used to break down centralized problems into distributed averaging sub tasks which provides flexibility but means network-wide convergence is required for each averaging process before the centralized equivalent algorithm may progress. It is therefore often better to use an approach designed from the ground up to deal with the problem in a distributed fashion, optimizing variables directly with each iteration, rather than trying to perform central optimization updates with global averages.

2.2.2 Belief Propagation

Belief propagation methods are designed within the framework of probability theory [90, 91, 134] and are used for performing statistical inference on probabilistic graphical models such as Markov random fields (MRF) [12], where a set of V memoryless random variables $\mathcal{X} = \{X_1, \ldots, X_V\}$ represent their dependencies through undirected edges (not so different, in fact, from the model of our wireless sensor network) [95, 145]. Note that for this subsection uppercase normal font letters such as X will refer to random variables. The goal is to determine the marginal distribution of each random variable within this set. The naive method of calculating this marginal distribution, in the case of discrete random variables taken from an alphabet $\mathcal{A}$, would be to sum over all possible configurations giving the marginal distribution of X_i

$$p_{X_i}(x_i) = \sum_{(X_j \in \mathcal{A}) \in \mathcal{X} \setminus \{X_i\}} p(x_1, \ldots, x_V), \tag{2.3}$$

where our summation is over all values of the random variable X_j from our alphabet $\mathcal{A}$ for all random variables in our set $\mathcal{X}$ apart from X_i. The time to calculate these is on the order of $|\mathcal{A}|^V$, which is obviously computationally prohibitive, but the structure of the graphical model may often be exploited to reduce this. One such case is when the factor graph representing our variables' probability mass function is a tree [12]. The

structure allows the summation of all configurations to now be expressed recursively by first summing over the leaves of the tree and progressing upwards to successive parent nodes. The time complexity of this procedure now grows linearly with V.

Distributed 'message passing' algorithms may be used to express the recursive summation where 'messages' associated with edges in the factor graph are used in local computations at graph nodes to produce iterative updates [75]. The update procedure that produces exact marginals in the case of tree structured graphs has been independently discovered in various fields such as probabilistic inference in Bayesian networks [104] and decoding theory [42], and is known by names such as the Bethe Peierls approximation, belief propagation, and the sum-product algorithm. Recently it has become popular due to its proposed effectiveness in more general graphs containing loops [139, 2]. The precision of the algorithm in these cases is related to the connectivity of the underlying factor graph since in extremely sparse situations we may view the graph locally as a tree. Since the algorithm performs local updates, intuitively this local tree structure should lead to success. The correlation of distant (in the sense of number of edges required to reach one another) variables will also degrade performance [2, 95].

Suppose we wish to compute the marginal distributions of continuous random variables in the set $\mathcal{X}$, where each random variable $X_i \in \mathcal{X}$ is associated with a single node $i \in \mathcal{N}$. The set of nodes $\mathcal{V}$ together with a set of undirected edges $\mathcal{E}$ form a MRF, assuming they satisfy the local Markov property (i.e. each node is conditionally independent of all other nodes given its neighbours). We may represent this MRF as a factor graph [12], which decomposes our joint density function as

$$p_{\mathcal{X}}(\boldsymbol{x}) = \prod_{i \in \mathcal{F}} f_i(\boldsymbol{x}_i), \tag{2.4}$$

where $p_{\mathcal{X}}$ is the joint density function, $\boldsymbol{x} = [x_1, \ldots, x_V]^T$ is the vector of our random variables, $\boldsymbol{x}_i$ is a subvector of $\boldsymbol{x}$ that forms a clique (or complete

subgraph) in the MRF, f_i is the potential (or factor) function associated with the clique, and $\mathcal{F}$ is the set of indexes for the potential functions. This density function may be further factorized into edge potentials ψ_{ij} and self potentials ϕ_i as [120]

$$p_{\mathcal{X}}(\boldsymbol{x}) \propto \prod_{i \in \mathcal{N}} \phi_i(x_i) \prod_{(i,j) \in \mathcal{E}} \psi_{ij}(x_i, x_j). \tag{2.5}$$

The edge potentials ψ_{ij} may be thought of as compatibility measures between the marginal estimates of nodes i and j while the self potentials ϕ_i may be thought of as evidence of the marginal estimate of node i. This leads to the BP update messages

$$m_{ij}(x_j) \propto \int_{x_i} \psi_{ij}(x_i, x_j) \phi_i(x_i) \prod_{l \in \mathcal{V}_i \setminus \{j\}} m_{li}(x_i) dx_i, \tag{2.6}$$

where $m_{ij}(x_j)$ is the message sent from node i to node j. We may then compute the marginals according to the product rule as [139, 120]

$$p_{X_i}(x_i) = \alpha \phi_i(x_i) \prod_{j \in \mathcal{N}_i} m_{ji}(x_i), \tag{2.7}$$

where $\alpha \in \mathbb{R}$ is a scaling parameter.

An important contribution in Gaussian belief propagation (where our marginal distributions are assumed to be Gaussian) very relevant to the work of this thesis is the link between the inference of the vector of marginal means using the message passing algorithm discussed above and the solution to the set of linear equations [120]

$$\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} = \boldsymbol{0}, \tag{2.8}$$

where $\boldsymbol{A} \in \mathbb{S}^N_{++}$ is a positive definite matrix, $\{\boldsymbol{b}, \boldsymbol{0}\} \in \mathbb{R}^N$ are column vectors, and $\boldsymbol{x} \in \mathbb{R}^N$ is the solution vector to this linear system. We may also solve this linear system by optimizing the convex quadratic program

$$\text{minimize} \quad \frac{1}{2}\boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}^T \boldsymbol{x}, \tag{2.9}$$

where the equivalence of these problems may be seen by simply setting the derivative of the quadratic function with respect to $\boldsymbol{x}$ equal to zero. Let us now define the joint Gaussian probability density function

$$p(\boldsymbol{x}) \triangleq Z^{-1} e^{-\frac{1}{2}\boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}^T \boldsymbol{x}}, \tag{2.10}$$

where Z is a distribution normalization factor. We denote $\boldsymbol{\mu} \triangleq \boldsymbol{A}^{-1}\boldsymbol{b}$ and may rewrite the Gaussian density function as

$$\begin{aligned} p(\boldsymbol{x}) &= Z^{-1} e^{\frac{1}{2}\boldsymbol{\mu}^T \boldsymbol{A}\boldsymbol{\mu}} e^{-\frac{1}{2}\boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} + \boldsymbol{u}^T \boldsymbol{A}\boldsymbol{x} - \frac{1}{2}\boldsymbol{u}^T \boldsymbol{A}\boldsymbol{u}} \\ &= \zeta^{-1} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{u})^T \boldsymbol{A}(\boldsymbol{x}-\boldsymbol{u})} \\ &= \mathcal{N}(\boldsymbol{u}, \boldsymbol{A}^{-1}), \end{aligned} \tag{2.11}$$

where $\zeta \triangleq Z e^{\frac{1}{2}\boldsymbol{u}^T \boldsymbol{A}\boldsymbol{u}}$ is the new normalization factor and $\mathcal{N}(\boldsymbol{u}, \boldsymbol{A}^{-1})$ is a multivariate normal distribution with mean vector $\boldsymbol{u}$ and covariance matrix $\boldsymbol{A}^{-1}$. Assuming non-neighbouring nodes in our graph are independent we may decompose this density function as in (2.6), distributing the nature of our cost function.

We may now solve the linear system (2.8) by inferring the marginal densities $p([\boldsymbol{x}]_i) \sim \mathcal{N}([\boldsymbol{u}]_i, [\boldsymbol{A}^{-1}]_{ii})$, where $[\cdot]_{ij}$ denotes the element of matrix $(\cdot)$ at row i and column j. In other words, we have performed a quadratic optimization distributed over V nodes where updates are performed using two parameters for each node - the mean and the variance. This can be extended to cost functions of higher order by using distributions with more parameters [91]. The Generalized Linear Coordinate-Descent (GLiCD) algorithm [152] generalizes this concept by providing an update scheme that requires only one parameter, regardless of the form of the individual cost functions, by incorporating feedback. In this manner we may perform general convex optimization across a graph using single parameter statistical inference. Belief propatation methods have found use in many applications, such as telecommunications (see [110, 93]).

2.2.3 Distributed Gradient Methods (DGM)

Distributed gradient methods tackle distributed processing by first defining local cost functions at each node which requires our original central function to be separable over all nodes. The global sum of these local costs is then minimized by the DGM iteratively. We begin by describing centralized subgradient ascent in the dual variable domain, followed by dual subgradient decomposition, and finally a primal gradient method known as diffusion adaptation.

Dual Gradient Ascent

Suppose we wish to minimize a convex function $f : \mathbb{R}^N \to \mathbb{R}$ subject to some linear constraint

$$\begin{aligned} &\text{minimize} \quad f(x), \\ &\text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \end{aligned} \tag{2.12}$$

where $\boldsymbol{x} \in \mathbb{R}^N$ is our primal optimization variable, and the matrix $\boldsymbol{A} \in \mathbb{R}^{M \times N}$ and vector $\boldsymbol{b} \in \mathbb{R}^M$ define a set of linear equality constraints. We form the Lagrangian, with multiplier $\boldsymbol{\nu} \in \mathbb{R}^M$, as

$$L(\boldsymbol{x}, \boldsymbol{\nu}) = f(\boldsymbol{x}) + \boldsymbol{\nu}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}), \tag{2.13}$$

from which we may derive the dual problem

$$\text{maximize} \quad g(\boldsymbol{\nu}), \tag{2.14}$$

where $g(\boldsymbol{\nu}) = \inf_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\nu})$ is the dual function [16]. Given that our original function f was convex and the problem only contained a linear equality constraint, Slater's condition (and therefore strong duality) will always hold allowing us to instead perform our optimization over the new dual problem. We may then recover the optimal primal point as

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\nu}^*), \tag{2.15}$$

where $\boldsymbol{x}^*$ and $\boldsymbol{\nu}^*$ are the optimal primal and dual variables, respectively, provided there is only one minimizer of $L(\boldsymbol{x}, \boldsymbol{\nu}^*)$ which can be ensured if f is strictly convex [15]. This dual problem form is often easier to solve since, firstly, it is now unconstrained and, secondly, if $M < N$ (i.e. there are fewer rows of linear constraints than the size of the primal variable $\boldsymbol{x}$) then the dual problem is an optimization of lower dimensionality than the primal problem. It is worth noting that the dual problem is also always convex, regardless of our original problem [16]. We can maximize $g(\boldsymbol{\nu})$ iteratively using gradient ascent

$$\boldsymbol{\nu}^{k+1} = \boldsymbol{\nu}^k + \mu \nabla g(\boldsymbol{\nu}_k), \tag{2.16}$$

where μ is the manually set step size and the gradient $\nabla g(\boldsymbol{\nu}^k)$ may be found as

$$\nabla g(\boldsymbol{\nu}^k) = \boldsymbol{A}\tilde{\boldsymbol{x}} - \boldsymbol{b} \tag{2.17}$$

where $\tilde{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\nu}^k)$. The dual ascent algorithm is therefore

$$\boldsymbol{x}^{k+1} = \arg\min_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\nu}^k), \tag{2.18}$$

$$\boldsymbol{\nu}^{k+1} = \boldsymbol{\nu}^k + \mu(\boldsymbol{A}\boldsymbol{x}^{k+1} - \boldsymbol{b}). \tag{2.19}$$

Dual Gradient Ascent Decomposition

Suppose that our cost function is separable such that $f(\boldsymbol{x}) = \sum_i f_i(\boldsymbol{x}_i)$ where $\boldsymbol{x}_i \in R^{V_i}$ are V disjoint subsets of $\boldsymbol{x}$. Note that we can always slice up the constraint matrix column-wise as $\boldsymbol{A}\boldsymbol{x} = \sum_{i \in \mathcal{V}} \boldsymbol{A}_i \boldsymbol{x}_i$. The Lagrangian is now separable across these subsets

$$L(\boldsymbol{x}, \boldsymbol{\nu}) = \sum_i L_i(\boldsymbol{x}_i, \boldsymbol{\nu}) \tag{2.20}$$

$$= \sum_i (f_i(\boldsymbol{x}_i) + \boldsymbol{\nu}^T \boldsymbol{A}_i \boldsymbol{x}_i - (1/V)\boldsymbol{\nu}^T \boldsymbol{b}), \tag{2.21}$$

and the $\boldsymbol{x}$-minimization in dual ascent is splittable into V separate minimizations

$$\boldsymbol{x}_i^{k+1} = \arg\min_{\boldsymbol{x}_i} L_i(\boldsymbol{x}_i, \boldsymbol{\nu}^k), \tag{2.22}$$

which can be computed in parallel. The basic dual decomposition (ascent) algorithm [35, 77, 46, 146] is then

$$\boldsymbol{x}_i^{k+1} = \arg\min_{\boldsymbol{x}_i} L_i(\boldsymbol{x}_i, \boldsymbol{\nu}^k) \quad \forall i, \tag{2.23}$$

$$\boldsymbol{\nu}^{k+1} = \boldsymbol{\nu}^k + \mu(\sum_{i=1}^{V} \boldsymbol{A}_i \boldsymbol{x}_i^{k+1} - \boldsymbol{b}). \tag{2.24}$$

As with standard dual gradient ascent, the algorithm requires f to be strictly convex in order to guarantee convergence. Practically for the algorithm to work $\boldsymbol{\nu}^k$ would be shared with all V nodes, each node would compute $\boldsymbol{x}_i^{k+1}$ in parallel, and all the constraint pieces $\boldsymbol{A}_i \boldsymbol{x}_i^{k+1}$ would be collected and used to update $\boldsymbol{\nu}^{k+1}$. Using this we are able to solve large problems through parallel subproblems.

The algorithm is simple and computationally efficient. However, its sublinear convergence rate of $O(1/k^{2/3})$ [66] is often slow in applications and a global summation or averaging phase is required at each dual update. This requires information to be mixed through the entire network and may be accomplished with simple distributed averaging algorithms such as the RGA.

Diffusion Adaptation

The adapt-then-combine diffusion strategy developed by Sayed *et al*. in [114] is an iterative procedure to optimize problems of the form

$$\text{minimize} \quad \sum_{k \in \mathcal{V}} f_k(\boldsymbol{x}), \tag{2.25}$$

where the cost functionals $f_k(\boldsymbol{x})$ are assumed to be convex and differentiable over $\boldsymbol{x}$, and are typically quadratic. The algorithm takes weighted averages of local gradients to form intermediate updates and then aver-

ages these for the final variable update. It may be expressed as

$$\begin{aligned} \tilde{\boldsymbol{x}}_i^k &= \boldsymbol{x}_i^k - \mu_i \sum_{j \in \mathcal{V}_i} c_{ji} [\nabla f_j(\boldsymbol{x}_i^k)], \\ \boldsymbol{x}_i^{k+1} &= \sum_{j \in \mathcal{V}_i} \boldsymbol{A}_{j|i} \tilde{\boldsymbol{x}}_j^k, \end{aligned} \tag{2.26}$$

where μ_i is a constant scalar step size, $\mathcal{V}_i$ represents the neighbourhood of node i, and $\boldsymbol{A}_{j|i}$ and c_{ji} are data flow coefficients from node j to node i chosen to satisfy

$$\begin{aligned} &\boldsymbol{A}_{j|i} \geq 0, \quad \sum_{i=1}^{V} \boldsymbol{A}_{j|i} = 1, \quad \boldsymbol{A}_{j|i} = 0 \text{ if } j \notin \mathcal{V}_i, \\ &c_{ji} \geq 0, \quad \sum_{i=1}^{V} c_{ji} = 1, \quad c_{ji} = 0 \text{ if } j \notin \mathcal{V}_i, \end{aligned} \tag{2.27}$$

to ensure these incremental update values produce a weighted average over the neighbourhood of node i.

The diffusion adaptation method differs from dual gradient ascent in three main ways. Firstly, the diffusion optimization variables will always tend towards equality across all nodes, i.e. $\boldsymbol{x}_1 = \cdots = \boldsymbol{x}_V$, since a common optimization variable $\boldsymbol{x}$ is assumed in the original problem. This is in contrast to the dual gradient ascent primal variables that are considered independent disjoint subsets of our original problem variable and may therefore take on values different from their neighbours. Secondly, diffusion is not able to deal with constraints directly whereas dual gradient ascent employs a linear matrix equality constraint $\sum_{i \in \mathcal{V}} \boldsymbol{A}_i \boldsymbol{x}_i = \boldsymbol{b}$. The final difference is the ability for diffusion to perform updates without a global collection phase, which is required for dual gradient ascent. This is the main strength of diffusion since it allows updates to be performed while new observations are being sampled by the sensor network, instead of requiring all nodes to wait until a satisfactory global averaging is complete.

The algorithm is simple to implement, relatively efficient computationally, and does not require a collection phase to reach convergence at

each time sample. However, it does not deal with constraints directly, requires synchronous gradient estimates to be performed across the whole network, and due to its gradient-based nature may practically never converge. The sometimes slow convergence of gradient methods is due to the nature of the gradient updates when the function being optimized has highly asymmetric level sets.

2.2.4 Alternating Direction Method of Multipliers (ADMM)

The alternating direction method of multipliers was developed in the 1970s [41, 48] and studied into 1980s and 1990s [36, 40, 39, 33, 25] but was mostly forgotten due to inadequate hardware for practical applications. It has since seen resurgence after a paper by Boyd *et al.* [15] represented the algorithm in the context of statistical machine learning with large data sets. The method was developed from dual ascent decomposition and the augmented Lagrangian method [57, 58, 11], also known as the method of multipliers, which also uses a quadratic penalty term to augment the Lagrangian function, as we will see.

We will focus on ADMM for general form consensus optimization where we suppose a theoretical global master node holds the variable $\boldsymbol{z} \in \mathbb{R}^V$. All nodes i contain a selection of the components of the master variable in their own local variable $\boldsymbol{x}_i \in \mathbb{R}^{V_i}$, where $V_i \leq V$. Unlike in the dual gradient ascent case, these components are not necessarily disjoint sets implying that multiple nodes may all have access to the same component of the global variable. We can describe this association with a mapping from the global variable index to local variable indices given by $g = G(i, j)$, meaning that global variable component $[\boldsymbol{z}]_g$ corresponds to local variable component $[\boldsymbol{x}_i]_j$. If we achieve consensus among our variables we have

$$[\boldsymbol{x}_i]_j = \boldsymbol{z}_{G(i,j)} \quad i = 1, \ldots, V, \quad j = 1, \ldots, N_i, \tag{2.28}$$

meaning that the master variable and all local variables are in agreement.

The general form consensus optimization problem is then

$$\begin{aligned} &\text{minimize} \quad \sum_{i\in\mathcal{V}} f_i(\boldsymbol{x}_i), \\ &\text{subject to} \quad \boldsymbol{x}_i - \tilde{\boldsymbol{z}}_i = 0 \quad \forall i \in \mathcal{V}, \end{aligned} \tag{2.29}$$

where $\tilde{\boldsymbol{z}}_i \in \mathbb{R}^{V_i}$ is defined by $[\tilde{\boldsymbol{z}}_i]_j = \boldsymbol{z}_{G(i,j)}$, i.e. $\tilde{\boldsymbol{z}}_i$ is what the global variable thinks $\boldsymbol{x}_i$ should be.

The augmented Lagrangian function for (2.29) is

$$L_\rho(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\nu}) = \sum_{i\in\mathcal{V}_i} \left(f_i(\boldsymbol{x}_i) + \boldsymbol{\nu}_i^T(\boldsymbol{x}_i - \tilde{\boldsymbol{z}}_i) + \frac{\rho}{2}\|\boldsymbol{x}_i - \tilde{\boldsymbol{z}}_i\|_2^2 \right) \tag{2.30}$$

where $\boldsymbol{\nu}_i \in \mathbb{R}^{V_i}$ is the dual variable and the final term is a quadratic penalty funtion, with the factor of $\frac{1}{2}$ present for algebraic simplicity. The penalty term is added to provide better practical convergence and robustness since any deviation from the linear consensus constraint is now explicitly penalized. Additionally, the inclusion of this term yields convergence without the assumption of strict convexity or finiteness of f, as was required in dual gradient ascent. Using alternating update steps, the ADMM Lagrangian is optimized as

$$\begin{aligned} \boldsymbol{x}_i^{k+1} &= \arg\min_{\boldsymbol{x}_i} \left(f_k(\boldsymbol{x}_i) + \boldsymbol{\nu}_i^{kT}\boldsymbol{x}_i + \frac{\rho}{2}\|\boldsymbol{x}_i - \tilde{\boldsymbol{z}}_i^k\|_2^2 \right), \\ \boldsymbol{z}_g^{k+1} &= (1/N_g) \sum_{G(i,j)=g} [\boldsymbol{x}_i^{k+1}]_j \\ \boldsymbol{\nu}_i^{k+1} &= \boldsymbol{\nu}_i^k + \rho(\boldsymbol{x}_i^{k+1} - \tilde{\boldsymbol{z}}_i^{k+1}). \end{aligned} \tag{2.31}$$

where N_g is the number of components corresponding to global index g. The first step $\boldsymbol{x}_i^{k+1}$ minimizes the primal variable over the augmented Lagrangian function, the second step $\boldsymbol{z}_g^{k+1}$ averages the components of all nodes corresponding to the global index g, and the final step $\boldsymbol{\nu}_i^{k+1}$ updates our dual variable using the difference between what our local variable is and what the theoretical master variable thinks it should be. Steps one and three may be computed independently and asynchronously while step

two requires an averaging phase over all nodes containing components corresponding to global index g. Whether this results in global averaging is dependent upon the problem and what each local variable is designed to hold.

ADMM functions well if a central collector or master node is present since the network-wide collection step is quickly processed. If there is no central collector then an averaging phase must be performed at each iteration which requires time to converge. In recent years, however, there has been a lot of work on reformulating the ADMM algorithm to remove the aggregation step in the dual update [138, 64] in order to make the algorithm more suited to distributed network based optimization. It has a linear convergence rate $O(1/k)$ [65], which is moderately faster than the gradient algorithms such as dual gradient ascent decomposition and diffusion adaptation. If the problem is formulated correctly then many affine equality constraints may be directly enforced while the independent nature of the iterations mean that asynchronous primal updates are possible. In some cases where the original cost functions are complicated the independent primal updates may be computationally expensive [15] leading to extensions such as Decentralized Linearized ADMM developed by Ling in [82]. In most cases, however, the primal update subproblems are efficiently solvable and often even have analytic solutions.

2.2.5 Primal-Dual Method of Multipliers (PDMM)

The Primal-Dual Method of Multipliers (PDMM) [158, 154], also referred to as the Bi-Alternating Direction Method of Multipliers due to the algorithm's formulation involving the convex bi-conjugate, draws inspiration from ADMM and uses similar Douglas-Rachford splitting on a primal-dual Lagrangian. While both methods use Gauss-Seidel iterations [50] to perform alternating primal and dual variable updates, the formulation and structure exploited by PDMM exhibits a faster convergence be-

haviour in certain scenarios when compared with ADMM [155] (although this may be primarily due to the parameter setting schemes of both methods) and provides a general framework for asynchronous and distributed updates when performing partial consensus optimization with general convex functions.

PDMM combines the practically rapid O(1/k) convergence rates of ADMM [138] with asynchronous and fully independent updates for closed, proper, and convex functions with general linear coupling constraints over each network edge, including partial consensus optimization [158, 127]. This is advantageous when optimizing a large number of variables, specifically in problems where only certain nodes contain information pertinent to certain variables. In a global consensus implementation of this type of problem all nodes would carry copies of all variables across the entire network, regardless of whether each node contributes to the optimization of these variables. The partial consensus approach is therefore more flexible for many physical network based problems where the number of variables increases with network size, and where neighbouring nodes contain information relevant to a common subset of global variables. However, the benefits of the PDMM algorithm come with added difficulty in deriving update equations for applications involving regularization, as computation of the Fenchel conjugate [16] of the original function, or optimization of coupled convex optimization problems, is required. This difficulty is compounded for many practical methods since the Fenchel conjugate of these regularized functions are often intractable without the necessary problem transformation, and the coupled optimization problems generally require numerical solution.

The partial consensus approach is of particular interest for problems where the total number of all distinct vector elements is significantly larger than the size of the local vectors $\boldsymbol{x}_i$ used at each node. In fact, in some physical network applications where nodes have access to overlapping vector elements and the number of distinct vector elements depends on

the size of the network, partial consensus becomes necessary for true scalability of network optimization. Problems that are of this form include distributed acoustic and image processing (e.g. beamforming [98], image fusion [99]) in large sensor networks.

PDMM, as presented in [156], optimizes the distributed general consensus problem

$$\begin{aligned} &\text{minimize} \quad \sum_{i\in\mathcal{V}} f_i(\boldsymbol{x}_i) \\ &\text{subject to} \quad \boldsymbol{A}_{i|j}\boldsymbol{x}_i = \boldsymbol{A}_{j|i}\boldsymbol{x}_j \quad \forall (i,j)\in\mathcal{E}, \end{aligned} \tag{2.32}$$

where f_i is assumed to be closed, proper and convex, and the two matrices $(\boldsymbol{A}_{i|j}, \boldsymbol{A}_{j|i})$ are known a priori for each edge $(i,j)\in\mathcal{E}$. A common example of the usage of edge-wise constraints in graphs is for partial consensus optimization, where equality is enforced between certain vector elements of neighbouring nodes. In this setting each local variable $\boldsymbol{x}_i$ may have elements in common with neighbouring nodes. In fact, by using edge-wise constraints we may enforce element-wise consensus between even distant nodes provided these nodes are connected by nodes also sharing the enforced elements. In other words, we assume that nodes with common elements form connected subgraphs. In this way, our general problem form (2.32) may be used to specify partial consensus optimization through edge-wise equality constraints, where the matrices $\{\boldsymbol{A}_{i|j}, \boldsymbol{A}_{j|i}\}$ contain entries of either 0, -1, or $+1$ (see [161, 98] for practical usefulness). It becomes a full consensus problem when the set of edge constraints are reduced to $\{\boldsymbol{x}_i = \boldsymbol{x}_j | (i,j)\in\mathcal{E}\}$. In the literature, the majority of research has focused on the full consensus problem [32, 96, 26, 123, 53, 85, 61], giving PDMM flexibility compared with these methods.

PDMM iteratively finds the saddle point of the augmented primal-dual Lagrangian for problem (2.32) defined as

$$L_\Phi(\boldsymbol{x},\boldsymbol{\lambda}) = \sum_{i\in\mathcal{V}} \Big[f_i(\boldsymbol{x_i}) - f_i^*\Big(\sum_{j\in\mathcal{V}_i} \boldsymbol{A}_{i|j}^T \boldsymbol{\lambda}_{i|j}\Big)$$

$$+ \sum_{j \in \mathcal{V}_i} \boldsymbol{\lambda}_{j|i}^T (\boldsymbol{A}_{i|j} \boldsymbol{x}_i) \Big] + \Phi(\boldsymbol{x}, \boldsymbol{\lambda}) \tag{2.33}$$

where $\boldsymbol{\lambda}_{i|j}$ is one of two symmetric dual variables for the edge (i, j) held by node i related to node j, f_i^* is the convex conjugate function of f_i [16], and $\Phi(\boldsymbol{x}, \boldsymbol{\lambda})$ is a primal-dual augmentation function defined as

$$\Phi(\boldsymbol{x}, \boldsymbol{\lambda}) = \sum_{(i,j) \in \mathcal{E}} \frac{1}{2} \Big[\rho \| \boldsymbol{A}_{i|j} \boldsymbol{x}_i - \boldsymbol{A}_{j|i} \boldsymbol{x}_j \|_2^2 - \frac{1}{\rho} \| \boldsymbol{\lambda}_{i|j} + \boldsymbol{\lambda}_{j|i} \|_2^2 \Big], \tag{2.34}$$

where ρ is a tuning parameter that determines the level of quadratic augmentation. The primal and dual variables are iteratively optimized for by asynchronously (or synchronously) triggering nodes for primal and dual update [156]. When a node i is randomly (with uniform distribution) selected for update, it performs

$$\boldsymbol{x}_i^{k+1} = \arg\min_{\boldsymbol{x}_i} \Big[f_i(\boldsymbol{x}_i) + \boldsymbol{x}_i^T \Big(\sum_{j \in \mathcal{V}_i} A_{ij}^T \boldsymbol{\lambda}_{j|i}^k \Big) + \sum_{j \in \mathcal{V}_i} \frac{\rho}{2} \| \boldsymbol{A}_{i|j} \boldsymbol{x}_i - \boldsymbol{A}_{j|i} \boldsymbol{x}_j^k \|_2^2 \Big], \tag{2.35a}$$

$$\boldsymbol{\lambda}_{i|j}^{k+1} = \rho (\boldsymbol{A}_{j|i} \boldsymbol{x}_j^k - \boldsymbol{A}_{i|j} \boldsymbol{x}_i^{k+1}) - \boldsymbol{\lambda}_{j|i}^k \quad \forall j \in \mathcal{V}_i, \tag{2.35b}$$

$$(\boldsymbol{x}_m^{k+1}, \boldsymbol{\lambda}_{m|q}^{k+1}) = (\boldsymbol{x}_m^k, \boldsymbol{\lambda}_{m|q}^k) \; \forall m \in \mathcal{V}, m \neq i, \forall q \in \mathcal{V}_m, \tag{2.35c}$$

where k is the update iteration number, (2.35c) indicates that all non-triggered node variables retain the value of their previous iteration, and the relation (2.35b) is used to update the dual variables so as not to require the potentially costly computation of the conjugate function f_i^* [158]. Additionally, synchronous operation is achieved by triggering all nodes simultenously for primal and dual variable update. The primal and dual updates, (2.35a) and (2.35b), require only local information in node i's neighbourhood from previous iterations k to perform independent updates. This means nodes may randomly and asynchronously perform updates, without the need for global synchronization or for a data sharing phase over more than a single neighbourhood.

Theoretical convergence analysis of synchronous and asynchronous PDMM is provided in [158] and [127] for decomposable convex functions. [158] makes use of variational inequality (VI) to conduct the analysis while [127] relies on the monotonic operator theory. In practice roughly equal update rates from uniformly distributed node triggers have been found to converge [98, 156]. The algorithm has been applied successfully for solving a number of practical problems, which include distributed dictionary learning [162], distributed support vector machine (SVM) [156], distributed speech enhancement over a wireless microphone network (see [130, 131, 98, 121]), and distributed image fusion [99].

In summary, PDMM exhibits good linear convergence rates of $\mathcal{O}(1/i)$ [159] that practically perform optimizations faster and more reliably than the gradient methods such as dual gradient ascent or diffusion. The PDMM algorithm can also sacrifice robustness to transmission error for communication efficiency by broadcasting variable updates to local neighbours [154] rather than sending node-specific updates. However, this broadcast implementation requires that no errors occur during transmission so may not be appropriate in very noisy or unstable networks. Some affine constraints may be enforced with manipulation of the original problem form but (as with many more sophisticated algorithms) PDMM requires more understanding, particularly in the realm of convex optimization, when manipulating the original problem.

2.3 Beamforming

Beamforming is a signal processing technique used in conjunction with an array of sensors or transmitters to accomplish spatial filtering [55, 133]. The beamformer exploits the principle of superposition of waves to estimate a signal from a desired direction in the presence of interferring signals and noise when multiple receivers are used, or to amplify a source signal when using multiple transmit antennas. This serves many practical

purposes, such as multiple transmission antennas in communication systems that boost signal transmit power in a chosen direction, or to reduce audio interference when using a microphone array for recording. Since a phase shift is equivalent to a time delay in the narrowband case (such as radar communication systems), receiver beamforming systems are able to weight and time delay the receive signals from each sensor [47]. These values are then summed to produce the beamformer output. In the wideband case, however, this time delay is no longer appropriate. The signals are therefore transformed to the frequency domain and phase shifted directly. This research will focus on the wideband case typical of audio systems.

We denote our wideband signal of interest as $s_0(t,\omega) \in \mathbb{C}$, with t and ω indexing the time sample and frequency subband, respectively. We may express the linear receive model at node i of our source signal, P interferers, and noise as

$$\begin{aligned} u_i(t,\omega) &= d_{i0}(\omega)s_0(t,\omega) + \sum_{p=1}^{P} d_{ip}(\omega)s_p(t,\omega) + v_i(t,\omega), \\ &= d_i(\omega)s(t,\omega) + n_i(t,\omega), \end{aligned} \tag{2.36}$$

where $u_i(t,\omega)$ is node k's observed noisy signal at time t and subband ω that is a combination of the desired signal $s_0(t,\omega)$, interferences $s_p(t,\omega)$ and additive noise $v_i(t,\omega)$. $d_{i0}(\omega)$ and $d_{ip}(\omega)$ represent the complex acoustic transfer functions that scale each subband of the source and interference signals, respectively, which are constant over time. The noise and interference may be combined into the variable $n_i(t,\omega)$ for convenience, allowing us to drop the 0 subscripts on $d_{i0}(\omega)s_0(t,\omega)$. The acoustic transfer function of node i at subband ω consists of an attenuation factor $a_i(\omega)$ and phase offset $e^{-j2\pi f_\omega \tau_i}$, where f_ω is the frequency associated with subband ω and τ_i is the relative delay from source to node i, i.e.

$$d_i(\omega) = a_i(\omega)e^{-j2\pi f_\omega \tau_i}. \tag{2.37}$$

When multiple sensors, such as microphones, are used we may represent the acoustic transfer functions to each sensor as the vector $\boldsymbol{d}(\omega) \in \mathbb{C}^N$,

allowing us to express a vector of noisy observations across these sensors as

$$\boldsymbol{u}(t,\omega) = \boldsymbol{d}(\omega)s(t,\omega) + \boldsymbol{n}(t,\omega), \tag{2.38}$$

where $\{\boldsymbol{u}(t,\omega), \boldsymbol{n}(t,\omega)\} \in \mathbb{C}^N$ are the vectors of observations and interferences from all N sensors, respectively. Beamforming aims to estimate the source signal $s(t,\omega)$ by combining these noisy observations in a weighted sum to produce a single scalar output

$$z(t,\omega) = \boldsymbol{w}^H(\omega)\boldsymbol{u}(t,\omega), \tag{2.39}$$

where $z(t,\omega)$ is the beamformed output and $\boldsymbol{w}(t,\omega) \in \mathbb{C}^N$ is the weighting vector used to combine the observations.

We will now describe three methods for determining the weight vector in a traditional centralized manner: the multichannel Wiener filter, which is optimal in the minimum mean square error (MMSE) sense; the Minimum Variance Distortionless Response (MVDR) beamformer, which sacrifices mean squared error performance for a distortionless source signal; and the simple delay-and-sum beamformer that is optimal in the presence of spatially uncorrelated noise.

2.3.1 Multichannel Wiener Filter

The multichannel Wiener filter was originally developed as a method of estimating a continuous process corrupted by additive noise [141, 140], and was then reformulated for the discrete time case [80] by Levinson. It provides a means of optimally estimating, in a minimum mean square error (MMSE) sense, our original source signal $s(t,\omega)$. We begin by expressing the error in each subband ω as $e(\omega) = s(\omega) - z(\omega)$, where we have assumed the signal and noise to be independent over time allowing us to omit the time index t. The squared error at time t may then be written as

$$\mathbf{E}[|\boldsymbol{e}(\omega)|^2] = \mathbf{E}[|s(\omega) - z(\omega)|^2] \tag{2.40}$$

$$= \mathbf{E}[|s(\omega) - \boldsymbol{w}^H(\omega)\boldsymbol{u}(\omega)|^2] \tag{2.41}$$

$$= \mathbf{E}[\big(s(\omega) - \boldsymbol{w}^H(\omega)\boldsymbol{u}(\omega)\big)\big(s^H(\omega) - \boldsymbol{u}^H(\omega)\boldsymbol{w}(\omega)\big)] \tag{2.42}$$

$$= r_{ss}(\omega) - \boldsymbol{w}^H(\omega)\boldsymbol{r}_{us}(\omega) - \boldsymbol{r}_{us}^H(\omega)\boldsymbol{w}(\omega) + \boldsymbol{w}^H(\omega)\boldsymbol{R}_{uu}(\omega)\boldsymbol{w}(\omega), \tag{2.43}$$

where $\mathbf{E}[\cdot]$ is the statistical expectated value operator, $r_{ss}(\omega)$ is the source signal autocorrelation, $r_{us}(\omega)$ is the cross-correlation of noisy observations $\boldsymbol{u}(\omega)$ and source signal $s(\omega)$, and $\boldsymbol{R}_{uu}(\omega)$ is the covariance matrix of the noisy observations. Assuming all subbands are independent, minimizing the sum of the error across all frequencies may be accomplished by minimizing each of these real-valued squared errors separately. Therefore

$$\nabla(\mathbf{E}[|\boldsymbol{e}(\omega)|^2]) = -2\boldsymbol{r}_{us}(\omega) + 2\boldsymbol{R}_{uu}(\omega)\boldsymbol{w}(\omega) = \mathbf{0}, \tag{2.44}$$

and, assuming the covariance matrix $\boldsymbol{R}_u u(\omega)$ is nonsingular, we may derive the optimal Wiener filter as [133, 17]

$$\boldsymbol{w}_{\mathrm{MMSE}}(\omega) = \boldsymbol{R}_{uu}^{-1}(\omega)\boldsymbol{r}_{us}(\omega), \tag{2.45}$$

where we have used the subscript $\boldsymbol{w}(\omega)_{\mathrm{MMSE}}$ to indicate that this is the optimal weight vector in the mean squared error sense. In practice the cross-correlation vector $\boldsymbol{r}_{us}(\omega)$ may be quite difficult to estimate as we don't necessarily have access to the clean source signal. This motivates the development of the MVDR beamformer, which is optimal in the mean squared error sense subject to no distortion of the source signal.

2.3.2 Minimum Variance Distortionless Response

The MVDR beamformer, also known as the Capon filter [20], does not require an estimate of the cross-correlation vector $\boldsymbol{r}_{us}(\omega)$ when determining a weight vector but instead requires knowledge of where the source is located. As such, the acoustic transfer function vector for each subband $\boldsymbol{d}(\omega)$ is assumed to be known. We now minimize the energy of the beamformed

signal $z(\omega)$ subject to unity (or distortionless) gain in the direction of the source. Any reduction in the beamformed signal's energy is therefore the result of attenuation of noise or interfering signals. This may be expressed as the constrained optimization problem

$$\begin{aligned} &\text{minimize} \quad \mathbf{E}[|\boldsymbol{w}^H(\omega)\boldsymbol{u}(\omega)|^2], \\ &\text{subject to} \quad \boldsymbol{d}^H(\omega)\boldsymbol{w}(\omega) = 1. \end{aligned} \tag{2.46}$$

Using the method of Lagrange multipliers it can be shown that the solution to this linearly constrained minimization problem in the presence of stationary and ergodic interference is [133, 17, 7]

$$\boldsymbol{w}_{\mathrm{MVDR}}(\omega) = \frac{\boldsymbol{R}_{uu}^{-1}(\omega)\boldsymbol{d}(\omega)}{\boldsymbol{d}^H(\omega)\boldsymbol{R}_{uu}^{-1}(\omega)\boldsymbol{d}(\omega)}. \tag{2.47}$$

This scheme requires knowledge of where the source is located but does not need an estimate of the cross-correlation $\boldsymbol{r}_{us}(\omega)$ for calculating the weight vector. We may explore the relation between the MMSE solution (2.45) and the MVDR solution (2.47) by first evaluating the autocorrelation of the output of the MVDR beamformer

$$r_{zz}^{\mathrm{MVDR}}(\omega) = \mathbf{E}[z_{\mathrm{MVDR}}(\omega)z_{\mathrm{MVDR}}^H(\omega)] \tag{2.48}$$

$$= \frac{\boldsymbol{d}^H(\omega)\boldsymbol{R}_{uu}^{-1}(\omega)\mathbf{E}[\boldsymbol{u}(\omega)\boldsymbol{u}^H(\omega)]\boldsymbol{R}_{uu}^{-1}(\omega)\boldsymbol{d}(\omega)}{\boldsymbol{d}^H(\omega)\boldsymbol{R}_{uu}^{-1}(\omega)\boldsymbol{d}(\omega)\boldsymbol{d}^H(\omega)\boldsymbol{R}_{uu}^{-1}(\omega)\boldsymbol{d}(\omega)} \tag{2.49}$$

$$= \frac{1}{\boldsymbol{d}^H(\omega)\boldsymbol{R}_{uu}^{-1}(\omega)\boldsymbol{d}(\omega)}. \tag{2.50}$$

Next, assuming that the source signal $s(\omega)$ and the noise vector $\boldsymbol{n}(\omega)$ are mutually independent, we may write the cross-correlation vector as

$$\boldsymbol{r}_{us}(\omega) = r_{ss}(\omega)\boldsymbol{d}(\omega). \tag{2.51}$$

Using (2.50) and (2.51) we may express the output of the Wiener filter as

$$z_{\mathrm{MMSE}}(\omega) = \boldsymbol{w}_{\mathrm{MMSE}}^H(\omega)\boldsymbol{u}(\omega) \tag{2.52}$$

$$= \frac{r_{ss}(\omega)}{r_{zz}^{\mathrm{MVDR}}(\omega)} \frac{\boldsymbol{d}^H(\omega)\boldsymbol{R}_{uu}^{-1}(\omega)\boldsymbol{u}(\omega)}{\boldsymbol{d}^H(\omega)\boldsymbol{R}_{uu}^{-1}(\omega)\boldsymbol{d}(\omega)} \tag{2.53}$$

$$= \frac{r_{ss}(\omega)}{r_{zz}^{\mathrm{MVDR}}(\omega)} z_{\mathrm{MVDR}}(\omega), \tag{2.54}$$

which is a product of the scalar mask $r_{ss}(\omega)/r_{zz}^{\mathrm{MVDR}}(\omega)$, called the Wiener post-filter [17, 7], and the MVDR output $z_{\mathrm{MVDR}}(\omega)$. The multichannel Wiener filter is therefore equal to the MVDR beamformer combined with a scalar post-filter. We may also view the MVDR beamformer as a specific case of the Wiener filter when the source signal autocorrelation and the MVDR output autocorrelation are equal. It should be noted that the signal-to-noise ratio (SNR) of the two methods is the same at each subband, however the Wiener filter results in better mean squared error than the MVDR beamformer. This is a result of the post-filter providing a complex scaling of the MVDR output, maintaining the SNR on each subband but reducing the contribution of particularly noisy subbands to the final signal output. The trade-off between the two approaches is lower mean squared error (and higher total signal SNR when averaged across all subbands) at the cost of higher frequency distortion to the source signal (since each subband is independently scaled).

In situations where estimation of the observation covariances are infeasible or difficult we may wish to implement the simpler delay-and-sum beamformer. This beamformer is optimal in the specific case of uncorrelated noise, as will be discussed next.

2.3.3 Delay-and-Sum Beamformer

The general delay-and-sum (GDS) beamformer may be derived from the MVDR beamformer if we assume the noise signals across all sensors are spatially uncorrelated with zero mean, as well as independent of the source signal. In this case the covariance matrix $\boldsymbol{R}_{uu} = \boldsymbol{R}_{dsds} + \boldsymbol{R}_{nn}$, where $\boldsymbol{R}_{dsds} = \mathbf{E}[|d_k(\omega)s(\omega)|^2]$ is the covariance of the noiseless attenuated source signal and $\boldsymbol{R}_{nn} = \mathbf{E}[|n_k(\omega)|^2] = \mathrm{diag}\{\sigma_{n_1}^2, \ldots, \sigma_{n_N}^2\}$ is the noise covariance matrix, which is diagonal due to spatial independence. Using the matrix

inversion lemma, it can be shown that the weight vector is now [17]

$$\boldsymbol{w}_{\mathrm{GDS}}(\omega) = \frac{\boldsymbol{R}_{nn}^{-1}(\omega)\boldsymbol{d}(\omega)}{\boldsymbol{d}^H(\omega)\boldsymbol{R}_{nn}^{-1}(\omega)\boldsymbol{d}(\omega)}, \tag{2.55}$$

which gives us the GDS beamformer output

$$\begin{aligned} z_{\mathrm{GDS}}(\omega) &= \boldsymbol{w}_{\mathrm{GDS}}(\omega)\boldsymbol{u}(\omega) && (2.56)\\ &= \frac{\sum_{i=1}^{V} a_i(\omega)(\sigma_{n_i}^2)^{-1}e^{j2\pi f_\omega \tau_i}u_i(\omega)}{\sum_{i=1}^{V} a_i^2(\omega)(\sigma_{n_i}^2)^{-1}} && (2.57)\\ &= \frac{\sum_{i=1}^{V} a_i^2(\omega)(\sigma_{n_i}^2)^{-1}\big(s(\omega) + a_i^{-1}(\omega)e^{j2\pi f_\omega \tau_i}n_i(\omega)\big)}{\sum_{i=1}^{V} a_i^2(\omega)(\sigma_{n_i}^2)^{-1}} && (2.58)\\ &= \frac{\sum_{i=1}^{V} b_i(\omega)\big(s(\omega) + \tilde{n}_i(\omega)\big)}{\sum_{i=1}^{V} b_i(\omega)} && (2.59)\\ &= s(\omega) + \frac{\sum_{i=1}^{V} b_i(\omega)\big(\tilde{n}_i(\omega)\big)}{\sum_{i=1}^{V} b_i(\omega)}, && (2.60) \end{aligned}$$

which is simply the original source signal $s(\omega)$ plus a weighted average of the scaled and phase shifted noise $\tilde{n}_i(\omega) = a_i^{-1}(\omega)e^{j2\pi f_\omega \tau_i}n_i(\omega)$ at each sensor, with weights $b_i(\omega) = a_i^2(\omega)(\sigma_{n_i}^2)^{-1}$. This method is optimal under the assumption that the noise signals at each node are spatially independent, leading to an output SNR of [17]

$$\mathrm{SNR}_{z_{GDS}} = \sum_{i=1}^{V} \mathrm{SNR}_i, \tag{2.61}$$

where $\mathrm{SNR}_{z_{GDS}}$ is the SNR of the GDS beamformer output and SNR_i is the SNR of node i's observation.

The basic delay-and-sum (DS) beamformer makes a further simplification of our system by assuming that the attenuation factors $a_i(\omega)$ are all unity and that the uncorrelated noise variances $\sigma_{n_i}(\omega)$ are all equal, i.e. $a_1(\omega) =, \ldots, = a_V(\omega) = 1$ and $\sigma_{n_1}(\omega) =, \ldots, = \sigma_{n_V}(\omega)$, respectively. This results in the simple beamformer

$$z_{\mathrm{DS}}(\omega) = \frac{1}{V}\sum_{i=1}^{V} e^{j2\pi f_\omega \tau_i}u_i(\omega) \tag{2.62}$$

where the observations $u_k(\omega)$ of each subband are first phase shifted (delayed) and then uniformly averaged (summed). This results in a collection of in-phase observations that is optimal if our conditions hold (unity attenuation factors; equal and uncorrelated noise). The output SNR in this specific case is therefore V times the SNR of each sensor [7]

$$\mathrm{SNR}_{z_{DS}} = V \cdot \mathrm{SNR}, \tag{2.63}$$

which is impressive considering the simplicity of the method.

2.3.4 Summary

We have found that for the general case of correlated noise with arbitrary complex acoustic transfer functions for each sensor, the MMSE optimal weight vector to estimate of our source signal is the Wiener filter. Given that the cross-correlation vector of the observations and the source signal is required for the Wiener filter, we have shown that the MVDR beamformer is an effective substitute that may be obtained without estimating this cross-correlation. In this case we instead require knowledge of the location of our source. The Wiener filter and MVDR beamformer produce outputs of equal SNR at each subband, with the latter trading some MMSE (and average signal SNR) performance for an undistorted source signal. Finally, we derive the GDS and the DS weight vectors from the MVDR weight vector by making progressively more restrictive system assumptions, for which these beamformer outputs are optimal. These each provide final output SNRs that are summations over the individual observation SNRs, given that our simplifying assumptions hold.

2.4 Existing Distributed Near-Field Beamformers

With the recent advancements in sensor networks [37, 72, 142, 115, 81] outlined previously, a natural progression for beamforming is its distribution over a set of self-contained nodes, each equipped with microprocessors, wireless EM communications and acoustic sensors. These nodes are assumed to be within communication range of only a local subset of the total nodes in the network. A distributed beamforming system should reduce total transmission energy required within the network, provide robustness to the addition or removal of nodes, eliminate the problem of a central master node failure, lower the coordination or calibration cost associated with setting up a large distributed network, and facilitate scalability of the beamformer to arbitrary network sizes [132].

One of the earliest distributed beamforming implementations for wireless sensor networks was by Bertrand and Moonen [8, 9] who distributed the processing of a linearly constrained minimum variance beamformer, which is a generalisation of the minimum variance distortionless response (MVDR) beamformer. Their method does not assume prior knowledge of the noise covariance matrix and can handle a full covariance matrix. However, their system does assume the network is fully connected (or connected in a tree topology, depending on the implementation method) and that each node k is equipped with multiple sensors. Additionally an ordering of the computations in the nodes is required.

The distributed delay-and-sum beamformer developed by Zeng and Hendriks [149] iteratively broadcasts information between neighbouring nodes using a randomized gossip protocol [14]. The approach requires no restriction on the network topology and may perform updates independently across all nodes. However, the noise at all nodes must be uncorrelated and assumed known, resulting in suboptimal performance in the presence of general correlated interference. The message-passing based

MVDR beamformer of [59] operates on scalar values in an asynchronous manner to perform weight vector optimization, does not require global convergence or collection phase for weight updates, and may be used in arbitrarily connected networks. However, the performance of the resulting MVDR beamformer is limited by the network topology as the interferences of non-neighbouring nodes are assumed to be uncorrelated. This is compounded by the requirement of the covariance matrix to be diagonally dominant, which is accomplished in [59] by an artificial damping of all off-diagonal elements. The diffusion-based MVDR beamformer of [97] is adaptive to varying interference statistics, only requires a single update per iteration, and approximates a full MVDR centralized beamformer with two-hop covariances. However, each node ultimately ends up with a large vector containing every node in the network's weight value as well as require the projection of this vector onto the linear constraint subspace. This limits the algorithm's true distributed nature, particularly in very large networks. Additionally, convergence can be slow due to the gradient optimization employed. All three of the above methods also require a global averaging at each time sample to produce a beamformer output.

More recently, the distributed privacy-protecting beamformer of [151] requires each node to hold a vector (with length on the order of the network size) of covariances with all other nodes, which is impractical and potentially impossible in very large networks. The LCMV beamformer of [10] assumes a fully connected network, but does contain details on how to extend the approach for more general networks. However, this extension requires node coordination to prune the partially connected network to a tree topology which introduces costly system overheads. Suboptimal approaches such as the time-frequency masking procedure in [128] and the related pseudo-coherence beamformer of [129] require less transmission power when compared with the above methods by utilizing sub-arrays of the total network. However, these approaches lack the flexibility to change their sub-array size depending on whether higher fidelity or lower power

consumption is desired. The three, primarily recursive, distributed LCMV beamformers reviewed in [87] are shown to be optimal when operating over a fully connected graph, and extension to partially connected network is briefly discussed. However, the partially connected network must first be pruned to a tree topology which incurs a computation overhead. Additionally, this approach is not robust to node removal or addition since any changes in the partially connected network may require another pruning operation to ensure a tree topology.

As there are relatively few distributed near-field acoustic beamformers currently in the literature, we suggest room for further development of these systems. We briefly outline a few of the current distributed beamformers as examples of how the distributed processing techniques in section 2.2 may be applied in practice. We cover: a linearly constrained minimum variance (LCMV) beamformer that is restricted to tree or fully connected networks; a delay-and-sum beamformer implemented with distributed averaging via RGA; an approximate MVDR beamformer that assumes interference covariance between only neighbouring nodes and utilizes a message passing algorithm for optimization; and finally, an approximate MVDR beamformer that assumes two hop interference covariance and uses the diffusion adaptation paradigm for weight vector optimization.

2.4.1 Distributed LCMV Beamforming in a Wireless Sensor Network With Single-Channel Per-Node Signal Transmission

As mentioned, one of the earliest distributed beamforming implementations for wireless sensor networks was by Bertrand and Moonen [8, 9] who distributed the processing of a linearly constrained minimum variance beamformer, which is a generalisation of the MVDR beamformer where the single MVDR linear constraint $\boldsymbol{d}^H\boldsymbol{w} = 1$ has been replaced with a more

general set of Q linear constraints given by $\boldsymbol{D}^H\boldsymbol{w} = \boldsymbol{f}$, where $\boldsymbol{D} \in \mathbb{C}^{N\times Q}$. The system assumes the network is fully connected (or connected in a tree topology, depending on the implementation method) and that each node i is equipped with M_i sensors.

Each node i is then able to produce a local beamformer output using these sensors as $z_i^k = \boldsymbol{w}_i^{kH}\boldsymbol{u}_i^k$, where $\boldsymbol{w}_i^k$, $\boldsymbol{u}_i^k \in \mathbb{C}^{M_i}$ are the weight vector and observation vector, respectively, for node i's sensors at iteration k. At each iteration every node broadcasts their local beamformer outputs to all other nodes to form the global beamformer output z^k

$$z^k = \boldsymbol{w}_i^{kH}\boldsymbol{u}_i^k + \sum_{j\in\mathcal{V}\ \{i\}} z_j^k, \tag{2.64}$$

followed by an update of each local weight vector $\boldsymbol{w}_i^k$ determined by a combination of the current local weight vector, the inverse of the local sensor covariance matrix $R_i^k \in \mathbb{C}^{M_i\times M_i}$, and the linear constraint $\boldsymbol{D}$. For a full description of the lengthy routine for the weight vector update see [9].

The algorithm broadcasts only scalar values and inverts only small local covariance matrices $\boldsymbol{R}_i^k$ when performing weight vector updates, resulting in a low total transmit cost when compared with the centralized computation procedure. The global covariances are also estimated implicitly sparing us the costly computation (and inversion) of a large network covariance matrix. The obvious limitations are that the network topology is restricted to either the fully connected case or to a tree structure, and that the updates must be performed synchronously across all nodes.

2.4.2 RGA delay and sum

The consensus based distributed delay-and-sum beamformer (DDSB) developed by Zeng and Hendriks [149] iteratively broadcasts information from a node k to one of its neighbours. A randomized gossip protocol is used to perform averaging of scalar values that converge on a beamformed output signal at each node per time sample. Since we are dealing with a

delay-and-sum beamformer, noise is assumed to be independent for each node k leading to a diagonal covariance matrix and allowing equations (8.3) and (2.47) to be combined to compute a beamformed signal as

$$z = \frac{\sum_{i=1}^{V} a_i \sigma_i^{-2} e^{-j\omega\tau_i} \boldsymbol{u}_i}{\sum_{i=1}^{V} a_i^2 \sigma_i^{-2}}, \tag{2.65}$$

where σ_i^2 is the noise variance at node i, and a_i and $e^{-j\omega\tau_i}$ represent the scalar magnitude scaling and phase offset, respectively, resulting from the transfer function d_i of node i. By assuming that each node i at time t has two initial values $\tilde{u}_i^0(t) = d_i^* \sigma_i^{-2} u_i(t) = a_i \sigma_i^{-2} e^{-j\omega\tau_i} u_i(t)$ and $\tilde{d}_i^0 = d_i^*(\sigma_i)^{-1} d_i = a_i^2(\sigma_i)^{-1}$ which are stacked into two V dimensional vectors $\tilde{\boldsymbol{u}}^0(t) = [\tilde{\boldsymbol{u}}_1^0(t), \cdots, \tilde{\boldsymbol{u}}_V^0(t)]^T$ and $\tilde{\boldsymbol{d}}^0 = [\tilde{d}_1^0, \cdots, \tilde{d}_V^0]^T$, equation (2.65) may be obtained as:

$$z = \tilde{u}_{\text{ave}} / \tilde{d}_{\text{ave}}, \tag{2.66}$$

where $\tilde{u}_{\text{ave}} = \frac{1}{V} \mathbf{1}^T \tilde{\boldsymbol{u}}^0(t)$ and $\tilde{d}_{ave} = \frac{1}{N} \mathbf{1}^T \tilde{\boldsymbol{d}}^0$ with $\mathbb{1}$ denoting an $N \times 1$ column vector of all ones. This is a ratio of network information averages and can therefore be iteratively calculated using a RGA by letting $\tilde{\boldsymbol{u}}^k(t)$ and $\tilde{d}^k$ be defined as vector $\tilde{\boldsymbol{u}}(t)$ and $\tilde{\boldsymbol{d}}$ at iteration k, respectively. Using equation (2.1), the state of these vectors at iteration k is given by

$$\begin{aligned} \tilde{\boldsymbol{u}}^{k+1}(t) &= U^{k+1} \tilde{\boldsymbol{u}}^k(t), \\ \tilde{\boldsymbol{d}}^{k+1} &= U^{k+1} \tilde{\boldsymbol{d}}^k, \end{aligned} \tag{2.67}$$

while the DDSB output of node i at time sample t and averaging iteration k is given by

$$\tilde{z}_i^k(t) = \tilde{u}_i^k(t) / \tilde{d}_i^k. \tag{2.68}$$

The assumption of known acoustic transfer function d_i and noise variance σ_i^2 is made since these may be estimated at each successive sample using methods such as [43] and [56], respectively, allowing attention to be focused on the distributed beamforming algorithm.

The distributed delay-and-sum beamformer is simple to implement, requires only scalar computations, and asynchronously shares scalar values with random neighbouring nodes meaning a low communication cost.

However, the algorithm requires peer-to-peer pairwise communications during updates and, more importantly, is limited by the diagonal assumption of the covariance matrix. This becomes especially pronounced in environments containing spatially occuring interferences such as a loud vehicle or another speaker.

2.4.3 Distributed MVDR Beamforming for (Wireless) Microphone Networks Using Message Passing

The distributed MVDR beamformer developed by Heusdens *et al.* [59] utilizes the GLiCD message-passing algorithm to compute an MVDR optimal estimate over a general sensor network. The GLiCD algorithm requires that all correlated nodes are connected in the graph, meaning that non-neighbouring nodes are assumed to have a covariance of zero. The beamformer uses message-passing to optimize a scaled weight vector $\tilde{\boldsymbol{w}}$ over all nodes via statistical inference, followed by a global averaging using RGA at each time sample to produce a beamformed signal output.

We begin by stating the centralized MVDR beamformer output as

$$z = \frac{\tilde{\boldsymbol{w}}^T \boldsymbol{u}}{\tilde{\boldsymbol{w}}^T \boldsymbol{d}}, \tag{2.69}$$

where $\tilde{\boldsymbol{w}} = \boldsymbol{R}^{-1}\boldsymbol{d}$. Next we assume that the covariance matrix has unit diagonal elements by rescaling our using $\boldsymbol{T} = \mathrm{diag}(\sqrt{[\boldsymbol{R}]_{11}}, \cdots, \sqrt{[\boldsymbol{R}]_{VV}})$. We therefore consider computing

$$\boldsymbol{x} = \boldsymbol{J}^{-1}\boldsymbol{h}, \tag{2.70}$$

where the unit-diagonal matrix $\boldsymbol{J} = \boldsymbol{T}^T\boldsymbol{R}\boldsymbol{T}$ and $\boldsymbol{h} = \boldsymbol{T}\boldsymbol{d}$. As described in section 2.2.2, we may relate this with a quadratic cost function

$$f(\boldsymbol{x}) \triangleq \frac{1}{2}\boldsymbol{x}^T\boldsymbol{J}\boldsymbol{x} - \boldsymbol{h}^T\boldsymbol{x} \tag{2.71}$$

which may be decomposed into pairwise cliques of the underlying graph $\mathcal{G}$ as

$$f(\boldsymbol{x}) = \sum_{i \in \mathcal{V}} f_i(\boldsymbol{x}_i) + \sum_{(i,j) \in \mathcal{E}} f_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j), \tag{2.72}$$

where f_i is a node potential function and f_{ij} is an edge potential function. This decomposition is possible since non-neighbouring nodes are assumed to have uncorrelated interference covariance. From [59, 152] it is shown that the update equations for the each

$$\begin{aligned} \boldsymbol{z}_{ij}^{k+1} = & \frac{\rho |[\boldsymbol{J}]_{ij}|^2}{1 - \rho^2 |[\boldsymbol{J}]_{ij}|^2} \left(\rho [\boldsymbol{h}]_j + \rho \sum_{m \in \mathcal{V}_j \setminus \{i\}} \boldsymbol{z}_{mj}^k + (1-\rho) \boldsymbol{x}_{j|i}^k \right) \\ & - \frac{\rho |[\boldsymbol{J}]_{ij}|^2}{1 - \rho^2 |[\boldsymbol{J}]_{ij}|^2} \left(\rho [\boldsymbol{h}]_i + \rho \sum_{n \in \mathcal{V}_i \setminus \{j\}} \boldsymbol{z}_{ni}^k + (1-\rho) \boldsymbol{x}_{i|j}^k \right), \end{aligned} \tag{2.73}$$

where $0 < \rho \leq 1$ controls the rate of convergence, z_{ij} is an intermediate variable for the edge (i, j), and

$$\boldsymbol{x}_{j|i}^{k+1} = [\boldsymbol{h}]_j + \sum_{m \in \mathcal{V}_j \setminus \{k\}} \boldsymbol{z}_{mj}^k + \boldsymbol{z}_{ij}^{k+1} \tag{2.74}$$

is the estimate of optimal element $[\boldsymbol{x}^*]_j$ held by node j relating to node i, which should converge to the same value for all neighbours i, i.e. $\boldsymbol{x}_{j|i}^k \to [\boldsymbol{x}^*]_j \; \forall i \in \mathcal{V}_j$ as $k \to \infty$.

The message passing-based MVDR beamformer operates on scalar values in an asynchronous manner to perform weight vector optimization and does not require a global convergence or collection phase at each iteration. However, the performance of the resulting MVDR beamformer is limited by the network topology as the interferences of non-neighbouring nodes are assumed to be uncorrelated. This is compounded by the requirement of the covariance matrix $\boldsymbol{R}$ to be diagonally dominant, which is accomplished in [59] by an artificial damping of all off-diagonal elements. Additionally, each node k is assumed to know the covariance with all of its neighbours.

For relatively small and fully connected networks this approach works well and distributes the calculation of the optimal weight vector. For large and fully connected networks knowledge of all neighbouring covariances may result in memory overload, while for large and sparsely connected networks the weight vector estimate falls short of the true MVDR performance as a result of the assumed independence of non-neighbouring nodes and the off-diagonal scaling.

2.4.4 Diffusion MVDR

The diffusion-based MVDR beamformer of [97] makes use of the diffusion adaptation algorithm for networks to iteratively optimize weight values across the sensors. The beamformer is partially MVDR since the algorithm assumes covariances only among at most two-hop neighbours leading to a sparse covariance matrix for network sensor observations where distant nodes are considered independent. The beamformer is therefore divided into two components: the weight vector optimization process; and the weighted averaging that is required to produce a beamformed output signal.

To determine the weight vector, local cost functions are first defined at each node i that include a prescaling term to account for multiple occurances of the same node to node covariances within the network. These are given by

$$J_k(\boldsymbol{w}) = \boldsymbol{C}^{\dagger 2} \circ \mathbf{E}[|\boldsymbol{u}_{\mathcal{V}_i}(t)^* \boldsymbol{w}|^2], \tag{2.75}$$

where $\boldsymbol{u}_{\mathcal{V}_i}(t)$ is the vector of observations in the neighbourhood of node i, $\circ$ is the Hadamard or element-wise product, and $\boldsymbol{C}^{\dagger 2}$ is the protected element-wise inverse of $\boldsymbol{C}^2$, the square of the adjacency matrix (also known as the binary connection matrix). This results in a global cost function given by the sum of all local costs

$$J^{\text{glob}}(\boldsymbol{w}) \triangleq \sum_{i=1}^{V} \boldsymbol{C}^{\dagger 2} \circ \mathbf{E}[|\boldsymbol{u}_{\mathcal{V}_i}(t)^* \boldsymbol{w}|^2] = \boldsymbol{w}^* \tilde{\boldsymbol{R}}_u \boldsymbol{w}, \tag{2.76}$$

where $\tilde{\boldsymbol{R}}_u$ is the partial covariance matrix of the network observations. The local cost function (2.75) is expanded and differentiated giving [89]

$$\begin{aligned} J_i(\boldsymbol{w}) &= \boldsymbol{w}^*(\boldsymbol{C}^{\dagger 2} \circ \tilde{\boldsymbol{R}}_u)\boldsymbol{w}, \\ -[\nabla J_i(\boldsymbol{w})]^* &= -(\boldsymbol{C}^{\dagger 2} \circ \tilde{\boldsymbol{R}}_u)\boldsymbol{w}, \end{aligned} \tag{2.77}$$

where an instantaneous approximation of this moment will be used

$$\tilde{\boldsymbol{R}}_u = \mathbb{E}[\boldsymbol{u}_{\mathcal{V}_i}(t)\boldsymbol{u}^*_{\mathcal{V}_i}(t)] \approx \boldsymbol{u}_{\mathcal{V}_i}(t)\boldsymbol{u}^*_{\mathcal{V}_i}(t). \tag{2.78}$$

With no linear constraint the trivial solution to equation (2.75) will be the zero vector, resulting in a beamformer output of zero. We will therefore orthogonally project the outputs of each diffusion iteration onto the subspace satisfying this linear constraint. This yields the following iterative diffusion process

$$\begin{aligned} \boldsymbol{\phi}_i^{k+1} &= \boldsymbol{w}_i^k - \mu_i \sum_{l \in \mathcal{N}_i} c_{ji}[\boldsymbol{C}^{\dagger 2} \circ \boldsymbol{u}_{\mathcal{V}_j}(t)\boldsymbol{u}^*_{\mathcal{V}_j}(t)]\boldsymbol{w}_i^k \\ \boldsymbol{\psi}_i^{k+1} &= \boldsymbol{P}_d^{\perp}\boldsymbol{\phi}_i^{k+1} + \boldsymbol{d}(\boldsymbol{d}^*\boldsymbol{d})^{-1} \\ \boldsymbol{w}_i^{k+1} &= \sum_{l \in \mathcal{V}_i} a_{ji}\boldsymbol{\psi}_j^{k+1} \end{aligned} \tag{2.79}$$

where $\boldsymbol{\phi}_i^{k+1}$ is a stochastic gradient descent step, $\boldsymbol{\psi}_i^{k+1}$ is an intermediate step that projects the estimate $\boldsymbol{\phi}_i^{k+1}$ onto the linear constraint subspace [16], and $\boldsymbol{w}_i^{k+1}$ is a summation of neighbourhood intermediate variables to diffuse information.

The diffusion process and projection requires knowledge of the network topology, a reasonable assumption due to recent developments [45], and also requires a transfer of receive vectors at the start of every iteration to form the vector $\boldsymbol{u}_{\mathcal{V}_i}(t)$. Once the weighting values are optimized a simple averaging process may be used to form the final beamformer output, such as RGA.

The diffusion-based MVDR beamformer is adaptive to varying interference statistics, only requires a single update per iteration, and approximates a full MVDR centralized beamformer with two-hop covariances.

However, each node will ultimately end up with a length N vector of every node in the network's weight value as well as require the projection of this vector onto the linear constraint subspace. This limits the algorithm's true distributed nature, particularly in very large networks. Additionally, convergence can be quite slow due to the gradient optimization steps and synchronous vector updates are required for each update iteration.

Chapter 3

A Distributed Beamformer using PDMM

In this chapter will describe the development of a distributed MVDR beamformer using both the alternating direction method of multipliers (ADMM) and the primal-dual method of multipliers (PDMM). We will first formulate our beamformer as a separable linearly constrained convex optimization problem in the primal domain and then derive the dual problem form. This will allow us to phrase the problem in a form acceptable by the ADMM and PDMM algorithm. We will derive update equations for a distributed ADMM MVDR beamformer and finally discuss the difficulty encountered in the PDMM case.

3.0.1 Derivation of a PDMM Beamformer

Recall that the traditional centralized MVDR beamformer minimizes output energy while maintaining a distortionless response in the direction of the source. This may be expressed as an optimization problem of the form

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \mathbf{E}[|\boldsymbol{u}^*\boldsymbol{w}|^2] = \boldsymbol{w}^*\boldsymbol{R}\boldsymbol{w}, \\ \text{subject to} \quad & \boldsymbol{d}^*\boldsymbol{w} = 1, \end{aligned} \tag{3.1}$$

where $\boldsymbol{u} \in \mathbb{C}^N$ is the complex observed signal vector, $\boldsymbol{w} \in \mathbb{C}^V$ is the complex weight vector over which the optimization is performed, $\boldsymbol{d} \in \mathbb{C}^V$ is the complex acoustic transfer function vector, and $\boldsymbol{R} \in \mathbb{C}_+^{V \times V}$ is the Hermetian positive semi-definite covariance matrix of the observed signals.

A problem that arises in optimization problems over complex variables is the difference in differentiability by real variables when compared with complex variables [76]. A common approach for dealing with this issue is to convert the optimization of real functions in complex variables into an optimization of real functions in real variables by considering the real and imaginary components of our complex variables as two separate real variables [18, 125]. This essentially doubles the dimension of our optimization problem as we are now optimizing over variables in $\mathbb{R}^{2V}$. However, these higher dimension expressions become cumbersome and only add to the confusion of our derivations. Therefore, throughout the rest of this report we will assume this transformation has already been performed and deal with real valued functions of dimension V, i.e. $\boldsymbol{u}, \boldsymbol{w}, \boldsymbol{d} \in \mathbb{R}^V$.

We now make the following three assumptions:

- **Assumption 1** - The underlying network is sparsely connected.

- **Assumption 2** - Observations of nodes not sharing a neighbourhood are assumed to be independent, i.e.

$$\mathbb{E}|\boldsymbol{u}_i^T \boldsymbol{u}_j| = [\boldsymbol{R}]_{i,j} = 0, \forall i, j \notin \mathcal{V}_k, \tag{3.2}$$

  where $[\boldsymbol{R}]_{i,j}$ is element (i, j) of the covariance matrix $\boldsymbol{R}$.

- **Assumption 3** - Each node k stores local estimates of neighbouring weight components and the estimated covariances of their neighbourhoods $[\boldsymbol{R}]_{i,j}\ \forall (i, j) \in \mathcal{V}_i$.

Using our system assumptions we may decompose the MVDR cost func-

tion as

$$\begin{aligned}\frac{1}{2}\boldsymbol{w}^*\boldsymbol{R}\boldsymbol{w} &= \frac{1}{2}\boldsymbol{w}^*(\boldsymbol{C}^{\dagger 2}\circ\boldsymbol{R})\boldsymbol{w} \\ &= \frac{1}{2}\sum_{i\in\mathcal{V}}\boldsymbol{w}_i^*\boldsymbol{R}_i\boldsymbol{w}_i\end{aligned} \tag{3.3}$$

where $\boldsymbol{w}_i \in \mathbb{R}^V$ is the vector of weights in the neighbourhood of node i, $C^{\dagger 2}$ is a mask that applies the network sparsity pattern and accounts for multiple neighbours carrying the same entry of the global weight vector $\boldsymbol{w}$, and $R_i \in \mathbb{S}_+^V$ is the resulting prescaled sparse covariance matrix of the neighbourhood $\mathcal{V}_i$. This has allowed us to separate our centralized cost function into a sum of N local cost functions present at each node i across the network.

A problem still remains, however, for application of PDMM. The global distortionless response constraint $\boldsymbol{d}^*\boldsymbol{w} = 1$ is not immediately able to be encoded in the edge-wise constraints present in the PDMM problem form (2.32). To deal with this problem we first recall the central analytic solution

$$\boldsymbol{w} = \frac{\boldsymbol{R}^{-1}\boldsymbol{d}}{\boldsymbol{d}^H\boldsymbol{R}^{-1}\boldsymbol{d}}. \tag{3.4}$$

Rather than attempting to find the weight vector $\boldsymbol{w}$ across all nodes we instead restrict our attention to finding a scaled version of the weight vector $\boldsymbol{x} = \boldsymbol{R}^{-1}\boldsymbol{d}$ since the beamformer output may then be calculated using gossip algorithms as

$$z = \frac{\boldsymbol{x}^T\boldsymbol{u}}{\boldsymbol{x}^T\boldsymbol{d}} = \frac{\frac{1}{V}\sum_{i\in\mathcal{V}}[\boldsymbol{x}]_i^T[\boldsymbol{u}]_i}{\frac{1}{V}\sum_{i\in\mathcal{V}}[\boldsymbol{x}]_i^T[\boldsymbol{d}]_i}. \tag{3.5}$$

This is simply the ratio of two global averages, which is required to produce a beamformer output signal regardless of the method used to optimize our weight vector. To obtain $\boldsymbol{x}$ we construct the unconstrained quadratic program

$$\text{minimize} \quad f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{R}\boldsymbol{x} - \boldsymbol{d}^T\boldsymbol{x}, \tag{3.6}$$

which we may then decompose as

$$\begin{aligned}\frac{1}{2}\boldsymbol{x}^T\boldsymbol{R}\boldsymbol{x} - \boldsymbol{d}^T\boldsymbol{x} &= \sum_{i\in\mathcal{V}}\left(\frac{1}{2}\boldsymbol{x}^T(\boldsymbol{C}^{\dagger 2}\circ\boldsymbol{R})\boldsymbol{x} - \boldsymbol{d}_i^T\boldsymbol{x}_i\right)\\ &= \sum_{i\in\mathcal{V}}\left(\frac{1}{2}\boldsymbol{x}_i^T\boldsymbol{R}_i\boldsymbol{x}_i - \boldsymbol{d}_i^T\boldsymbol{x}_i\right).\end{aligned} \tag{3.7}$$

where $\boldsymbol{x}_i \in \mathbb{R}^{V_i}$ is the local vector of neighbourhood elements from the global vector $\boldsymbol{x}$, $\boldsymbol{d}_i \in \mathbb{R}^{V_i}$ is a vector containing all zeros apart from the kth entry which is equal to node i's scalar ATF d_i, and $\boldsymbol{R}_i \in \mathbb{R}^{V_i\times V_i}$ is the covariance matrix for the neighbourhood $\mathcal{V}_i$. Provided we enforce pairwise consensus constraints between the primary scalar element of node i (i.e. $[\boldsymbol{x}]_i$) and the copy held by node j for all $j \in \mathcal{V}_i$, we obtain the optimal vector entries of $\boldsymbol{x}$ across neighbourhoods of k as solutions of

$$\begin{aligned}\text{minimize}\quad & \sum_{i\in\mathcal{V}} f_i(\boldsymbol{x}_k) = \sum_{i\in\mathcal{V}}\left(\frac{1}{2}\boldsymbol{x}_i^T\boldsymbol{R}_i\boldsymbol{x}_i - \boldsymbol{d}_i^T\boldsymbol{x}_i\right),\\ \text{subject to}\quad & \boldsymbol{A}_{i|j}\boldsymbol{x}_i = \boldsymbol{A}_{j|i}\boldsymbol{x}_j \quad \forall (i,j)\in\mathcal{E},\end{aligned} \tag{3.8}$$

where the matrices $\boldsymbol{A}_{i|j} \in \mathbb{R}^{2\times V_i}$ and $\boldsymbol{A}_{j|i} \in \mathbb{R}^{2\times V_j}$ contain entries of 1 or 0 to enforce consistency (with one row each for the consensus of node i and j's primary elements and their copies), and the local vector $\boldsymbol{x}_i$ is node ik's idea of the elements of $\boldsymbol{x}$ belonging to its neighbourhood $\mathcal{V}_i$. Using the PDMM update equations of section 2.2.5 we arrive at the iterations $\forall i \in \mathcal{V}$

$$\boldsymbol{x}_i^{k+1} = \Big(\sum_{j\in\mathcal{V}_i}\rho\boldsymbol{A}_{i|j}^T\boldsymbol{A}_{i|j} + \boldsymbol{R}_i\Big)^{-1}\Big(\boldsymbol{d}_i + \sum_{j\in\mathcal{V}_i}\boldsymbol{A}_{i|j}^T(\rho\boldsymbol{A}_{j|i}\boldsymbol{x}_j^k - \boldsymbol{\lambda}_{j|i}^k)\Big) \tag{3.9}$$

$$\boldsymbol{\lambda}_{i|j}^{k+1} = \rho(\boldsymbol{A}_{j|i}\boldsymbol{x}_j^k - \boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1}) - \boldsymbol{\lambda}_{j|i}^k \qquad \forall j \in \mathcal{V}_i. \tag{3.10}$$

In the next section we will test the proposed PDMM beamformer in a simulated environment to confirm that it converges to the central solution obtained by (3.4). Recall that this approach does not include any form of weight regularization, such as attempting to reduce the energy of weights or induce sparsity in our weight set.

3.1 Numerical Simulation

A wireless sensor network was simulated with $V = 20$ microphone nodes and a point source randomly distributed in a $10\,\mathrm{m} \times 10\,\mathrm{m} \times 10\,\mathrm{m}$ room. The distances from node k to all its neighbours were assumed to be known and the network was connected in a random sparse manner corresponding to a randomly generated sparse covariance matrix. Edges exist between nodes k and l if there is a nonzero covariance entry at row k and column l of the covariance matrix. The acoustic transfer function for each node is a complex scaling by $d_k = \frac{1}{l_k} e^{-\mathrm{j}\tau_k}$ where l_k is the distance between node k and the source, $\tau_k = \frac{l_k}{c} 2\pi f_s$ and the speed of sound $c = 340\ \mathrm{ms}^{-1}$.

The signal of interest was a 10 second speech sample randomly chosen from a 60 second recording and the interference was zero-mean Gaussian using the sparse covariance matrix as described above. The covariance matrix was scaled by the speech sample's power to produce a signal-to-noise ratio (SNR) of either 5 dB or -5 dB, followed by a transfer function scaling to produce distant dependent covariances. This covariance matrix was then used to create interference for each node. In this way the observed signal at each node was the sum of the source signal and some random correlated interference, scaled by the respective distance dependent transfer functions, allowing both the SNR and covariances to be perfectly determined. The sample rate at each node was $f_s = 8$ kHz and processing was carried out on 6.25 ms Hanning windowed blocks with a 50% overlap.

We show the convergence of the PDMM weight vector to the MVDR optimal weight vector as a function of update iterations. Weight vector mean squared error (MSE) of the PDMM estimated weight vector at node i compared with the MVDR optimal weight vector, averaged over Q realisations, is defined as

$$\mathrm{MSE}_i^k = \frac{1}{Q} \sum_{q=1}^{Q} \|\boldsymbol{w}_{q,i}^k - \boldsymbol{w}_{q,\mathrm{MVDR}}\|^2 \tag{3.11}$$

where $w_{q,k}^i$ and $w_{q,\mathrm{MVDR}}$ are the estimated weight vector of node k at iter-

ation i and the optimal weight vector, respectively, for the qth realisation. This is the most fundamental measure of our algorithms convergence since the optimization of our weight vector in the mean squared error sense is what we are attempting to achieve.

3.1.1 Results and Discussion

We now present the simulation results of our PDMM beamformer when compared with the MVDR optimal beamformer as discussed in section 3.0.1. The weight vector convergence for noisy observations SNRs of 5 dB, 0 dB and -5 dB were simulated and the results are shown in figure 3.1. We see that the weight vector error follows the same, sublinear, convergence rate regardless of the noisy observation SNRs. This is consistent with the convergence properties discussed in section 2.2.5. The PDMM

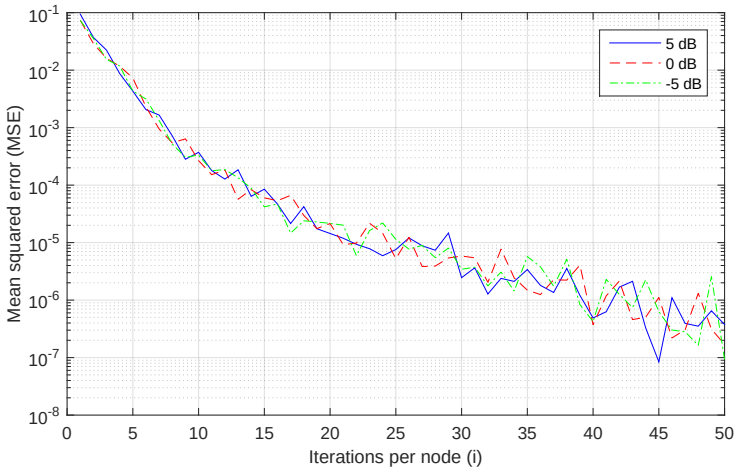


Figure 3.1: MSE of PDMM weight vector relative to MVDR weight vector as a function of iteration number

beamformer converges on the optimal MVDR beamformer weight vector

given our initial assumption that nodes not sharing a common neighbourhood experience uncorrelated interference. This is the same assumption made in the diffusion-based MVDR beamformer summarized in section 2.4.4. In sparsely connected networks this is appropriate but in densely connected large networks this may not be the case.

Suppose now that we wish to apply a form of regularization to the weight vector $\boldsymbol{w}$ and therefore the optimization vector $\boldsymbol{x}$. A sparsity inducing regularizer such as the l_1 norm [5, 117] to reduce the number of active distributed weight values. This may be desirable as it would reduce the number of total nodes within the network that must transmit their scaled signal observations for beamformer output. For a general regularization function R_i at each node i, our distributed optimization problem would then be

$$\begin{aligned} &\text{minimize} \quad \sum_{i\in\mathcal{V}} \left(\frac{1}{2}\boldsymbol{x}_i^T \boldsymbol{R}_i \boldsymbol{x}_i - \boldsymbol{d}_i^T \boldsymbol{x}_i + R_i(\boldsymbol{x}_i) \right), \\ &\text{subject to} \quad \boldsymbol{A}_{i|j}\boldsymbol{x}_i = \boldsymbol{A}_{j|i}\boldsymbol{x}_j \quad \forall (i,j) \in \mathcal{E}, \end{aligned} \tag{3.12}$$

leading to the PDMM update iterations

$$\begin{aligned} \boldsymbol{x}_i^{k+1} = \arg\min_{\boldsymbol{x}_i} \Bigg(&\frac{1}{2}\boldsymbol{x}_i^T \boldsymbol{R}_i \boldsymbol{x}_i - \boldsymbol{d}_i^T \boldsymbol{x}_i + \boldsymbol{R}_i(\boldsymbol{x}_i) \\ &+ \boldsymbol{x}_i^T \Big(\sum_{j\in\mathcal{V}_i} \boldsymbol{A}_{i|j}^T \boldsymbol{\lambda}_{j|i}^k \Big) + \sum_{j\in\mathcal{V}_i} \frac{\rho}{2} \|\boldsymbol{A}_{i|j}\boldsymbol{x}_i - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^k\|_2^2 \Bigg), \end{aligned} \tag{3.13}$$

$$\boldsymbol{\lambda}_{i|j}^{k+1} = \rho(\boldsymbol{A}_{j|i}\boldsymbol{x}_j^k - \boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1}) - \boldsymbol{\lambda}_{j|i}^k \qquad \forall j \in \mathcal{V}_i. \tag{3.14}$$

The primal variable update (3.13) may require a costly numerical solution at each update iteration k, such as when our regulization function is the l_1 norm. This presents a real problem for distributed sensor networks, where computation power and battery life are very limited. Additionally, small sensors may not be equipped with the optimization packages necessary for this numerical solution as available memory is also a limited resource. In the following chapter, we will investigate an approach for splitting the cost and regularization functions with the aim of reducing the computational cost of distributed PDMM regularized optimization.

Chapter 4

PDMM Network Function Splitting

Motivated by the complications that arise when employing regularized local subproblems for distributed PDMM optimization, in this chapter we develop a variation of PDMM that allows for the separation of local functions, which we refer to as Function Splitting PDMM (FS-PDMM). We show that this approach simplifies local update iterations that must be solved for each node, at each update iteration, improving computational efficiency at the network's distributed processors. In particular, the local updates become greatly simplified when using regularization functions with closed-form proximal operator expressions, such as common l_1 and l_2 norm penalties used in Tikhonov regularization [49], sparse l_1 regularization [116, 78], LASSO [88, 38], and elastic net regularization [163].

We begin by presenting the general regularized problem form assumed throughout this chapter and show how we might naïvely split the cost and regularization functions using a separate application of PDMM or ADMM, followed by the FS-PDMM algorithm. An equivalence analysis is then performed to prove that FS-PDMM is theoretically equivalent to performing conventional PDMM on a network twice the size as the physical network. Finally, we presented simulated experiments that confirm the effectiveness

of FS-PDMM when applied to an elastic net [163] regularized least-squares problem.

4.1 Function Splitting

In this section we address the difficulty of computing local PDMM primal variable updates when using regularized functions, and describe a simple method of local function splitting that requires numerical solution at each update iteration. We then present an asynchronous updating proceedure that separates the cost functions f_i and regularization functions R_i across two local subproblems. We show that this separation is particularly useful when our regularization functions have simple proximal operator expressions, such as for common l_1 and l_2 norms.

4.1.1 Problem Form and Naïve Approach

To begin, we assume that over our network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ we wish to perform regularized optimization of the form

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{V} \big(f_i(\boldsymbol{x}_i) + R_i(\boldsymbol{M}_i \boldsymbol{x}_i) \big) \\ \text{subject to} \quad & \boldsymbol{A}_{i|j} \boldsymbol{x}_i = \boldsymbol{A}_{j|i} \boldsymbol{x}_j \quad \forall (i,j) \in \mathcal{E}, \end{aligned} \tag{4.1}$$

where $\boldsymbol{M}_i \in \mathbb{R}^{\boldsymbol{M}_i \times n_i}$ defines a general linear mapping on the variable $\boldsymbol{x}_i$ prior to regularization, the loss function $f_i : \mathbb{R}^{n_i} \to \mathbb{R} \cup \{+\infty\}$ and regularization function $R_i : \mathbb{R}^{\boldsymbol{M}_i} \to \mathbb{R} \cup \{+\infty\}$ at node i are both closed, proper, and convex, the matrices $\{\boldsymbol{A}_{i|j}, \boldsymbol{A}_{j|i}\}$ define a linear constraint across the edge (i, j). Compared with the general PDMM problem form (2.32), here we have the local functions $g_i(\boldsymbol{x}_i) = f_i(\boldsymbol{x}_i) + R_i(\boldsymbol{M}_i \boldsymbol{x}_i)\ \forall i \in \mathcal{V}$.

When g_i in problem (2.32) is a relatively simple function, such as a quadratic, updating the primal variable requires a minimization that may be computed analytically and efficiently. However, when applying PDMM

to problem (4.1), performing the PDMM primal variable update requires an iterative numerical solution of each local subproblem. In low cost, large sensor networks this iterative minimization at each node may be prohibitive, both in terms of computation power at each node and given the potentially limited library of numerical solvers available to each sensor.

To solve the PDMM primal update (2.35a) for a regularized problem of the form (4.1), perhaps the most obvious approach would be to reformulate each local subproblem with the auxilliary variable $\boldsymbol{z}_i = \boldsymbol{M}_i\boldsymbol{x}_i$. The PDMM primal variable iterate $\boldsymbol{x}_i^{k+1}$ at node i may then be found as the solution to the constrained optimization problem

$$\begin{aligned} \text{minimize} \quad & \Bigg[f_i(\boldsymbol{x}_i) + \sum_{j\in\mathcal{V}_i} \frac{\delta}{2}\|\boldsymbol{A}_{i|j}\boldsymbol{x}_i - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^k\|_2^2 \\ & \qquad + \boldsymbol{x}_i^T\Big(\sum_{j\in\mathcal{V}_i} \boldsymbol{A}_{i|j}^T\boldsymbol{\lambda}_{j|i}^k\Big)\Bigg] + R_i(\boldsymbol{z}_i) \\ \text{subject to} \quad & \boldsymbol{z}_i = \boldsymbol{M}_i\boldsymbol{x}_i. \end{aligned} \tag{4.2}$$

Applying ADMM [15] directly to this problem, leads to the iterates

$$\begin{aligned} \boldsymbol{x}_i^{\kappa+1} = \arg\min_{\boldsymbol{x}_i} \Bigg[& f_i(\boldsymbol{x}_i) + \boldsymbol{x}_i^T\Big(\sum_{j\in\mathcal{V}_i} \boldsymbol{A}_{i|j}^T\boldsymbol{\lambda}_{j|i}^k + \boldsymbol{M}_i^T\boldsymbol{\nu}_i^\kappa\Big) \\ & + \sum_{j\in\mathcal{V}_i} \frac{\rho}{2}\|\boldsymbol{A}_{i|j}\boldsymbol{x}_i - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^k\|_2^2 + \frac{\delta}{2}\|\boldsymbol{M}_i\boldsymbol{x}_i - \boldsymbol{z}_i^\kappa\|_2^2\Bigg] \end{aligned} \tag{4.3a}$$

$$\boldsymbol{z}_i^{\kappa+1} = \arg\min_{\boldsymbol{z}_i} \left[R_i(\boldsymbol{z}_i) + \frac{\delta}{2}\|\boldsymbol{z}_i - (\boldsymbol{M}_i\boldsymbol{x}_i^{\kappa+1} + \boldsymbol{\nu}_i^\kappa/\delta)\|_2^2 \right] \tag{4.3b}$$

$$\boldsymbol{\nu}_i^{\kappa+1} = \boldsymbol{\nu}_i^\kappa + \delta(\boldsymbol{M}_i\boldsymbol{x}_i^{\kappa+1} - \boldsymbol{z}_i^{\kappa+1}), \tag{4.3c}$$

where we are using κ to distinguish the local ADMM iterates from the current global PDMM iteration index k, and δ controls the level of quadratic augmentation for ADMM.

Taking the above approach means that at each global PDMM iterate k we need to repeat (4.3a)-(4.3c) until we obtain an estimate to the solution of the PDMM primal subproblem (2.35a). Additionally, for complex cost

functions f_i the ADMM iterate (4.3a) may itself require numerical solution, adding yet another nested loop to our network optimization.

4.1.2 Function Splitting and Proximal Operators

Given our regularized problem (4.1), we propose the asynchronous Function Splitting PDMM (FS-PDMM) updating scheme given in Algorithm 1, where we exploit the proximal operator [94, 103, 29], defined for a function h and vector $\boldsymbol{v}$ as

$$\mathbf{prox}_{h,1/\rho}(\boldsymbol{v}) = \arg\min_{x} \left(h(\boldsymbol{x}) + \frac{\rho}{2}\|\boldsymbol{x} - \boldsymbol{v}\|_2^2 \right) \tag{4.4}$$

to simplify update expressions. We now have an auxiliary primal variable $\boldsymbol{z}_i$ and dual variable $\boldsymbol{\nu}_i$ at each node i, and perform updates very similar to a *single pass* over the ADMM updates (4.3a)-(4.3c). We will see in Section 4.2 that these updates may be interpreted as PDMM iterates performed at auxiliary "virtual" nodes attached to each physical node.

The first primal update of Algorithm 1 requires the minimization of the cost function f_i and a quadratic penalty over the primal variable $\boldsymbol{x}_i$, while the second primal update requires the minimization of the regularization function R_i and a quadratic penalty (equivalently, the proximal update) over the primal variable $\boldsymbol{z}_i$. The final two dual variable updates are simple linear mappings. The proximal update may be easily evaluated for many common regularization functions, such as the l_1 and l_2 norms, leaving us with only a single cost function minimization. In contrast, using conventional PDMM would require performing a coupled minimization over both functions f_i and R_i as well as a quadratic penalty, which generally would require numerical optimization. The local computational savings of FS-PDMM will be demonstrated in Section 4.3.

We show in Section 4.2 that FS-PDMM is, in fact, equivalent to conventional PDMM performed over a network of $2V$ nodes connected in a very similar topology to our physical network. We can therefore apply all

Algorithm 1 FS-PDMM

Initialize:

specify $f_i, R_i, \boldsymbol{M}_i, \boldsymbol{A}_{i|j}, \rho, \epsilon \qquad \forall i \in \mathcal{V}, \forall j \in \mathcal{V}_i$

randomly initialize $(\boldsymbol{x}_i, \boldsymbol{z}_i, \boldsymbol{\nu}_i, \boldsymbol{\lambda}_{i|j})\ \forall i \in \mathcal{V}, \forall j \in \mathcal{V}_i$

Update procedure:

while *stopping criterion* $==$ *false* **do**

  randomly trigger node i for update

  $$\boldsymbol{x}_i^{k+1} = \arg\min_{\boldsymbol{x}_i} \Bigg[\frac{\rho}{2} \left\| \boldsymbol{M}_i \boldsymbol{x}_i - \boldsymbol{z}_i^k \right\|_2^2 + f_i(\boldsymbol{x}_i) + \boldsymbol{x}_i^T \Big(\textstyle\sum_{j \in \mathcal{V}_i} \boldsymbol{A}_{i|j}^T \boldsymbol{\lambda}_{j|i}^k + \boldsymbol{M}_i^T \boldsymbol{\nu}_i^k \Big) + \textstyle\sum_{j \in \mathcal{V}_i} \frac{\rho}{2} \left\| \boldsymbol{A}_{i|j} \boldsymbol{x}_i - \boldsymbol{A}_{j|i} \boldsymbol{x}_j^k \right\|_2^2 \Bigg]$$

  $$\boldsymbol{z}_i^{k+1} = \mathbf{prox}_{R_i, 1/\rho}(2\boldsymbol{M}_i \boldsymbol{x}_i^{k+1} - \boldsymbol{z}_i^k + \boldsymbol{\nu}_i^k / \rho)$$

  $$\boldsymbol{\nu}_i^{k+1} = \rho(2\boldsymbol{M}_i \boldsymbol{x}_i^{k+1} - \boldsymbol{z}_i^{k+1} - \boldsymbol{z}_i^k) + \boldsymbol{\nu}_i^k$$

  $$\boldsymbol{\lambda}_{i|j}^{k+1} = \rho(\boldsymbol{A}_{j|i} \boldsymbol{x}_j^k - \boldsymbol{A}_{i|j} \boldsymbol{x}_i^{k+1}) - \boldsymbol{\lambda}_{j|i}^k \qquad \forall j \in \mathcal{V}_i$$

  share $\boldsymbol{x}_i^{k+1}, \boldsymbol{\lambda}_{i|j}^{k+1}$ with neighbours $j \in \mathcal{V}_i$

  for $m \in \mathcal{V}, m \neq i, \forall q \in \mathcal{V}_m$ **do**

    $(\boldsymbol{x}_m^{k+1}, \boldsymbol{z}_m^{k+1}, \boldsymbol{\nu}_m^{k+1}, \boldsymbol{\lambda}_{m|q}^{k+1}) = (\boldsymbol{x}_m^k, \boldsymbol{z}_m^k, \boldsymbol{\nu}_m^k, \boldsymbol{\lambda}_{m|q}^k)$

  end for

  if $\|\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k\| + 1/V_i \sum_{j \in \mathcal{V}_i} \|\boldsymbol{\lambda}_{i|j}^{k+1} - \boldsymbol{\lambda}_{i|j}^k\| < \epsilon$ **then**

    stopping criterion $=$ *true*

  end if

  $k \leftarrow k + 1$

end while

PDMM [158] results, including the stopping criterion based on primal and variable estimate stability and the O(1/k) convergence for cyclically triggered node updates, to this transformed network topology. In Section 4.3 we apply FS-PDMM to an elastic net (linearly combined l_1 and l_2) regularized least-squares problem.

4.2 Equivalence Analysis

In this section we analyze the FS-PDMM algorithm and show the equivalence of FS-PDMM to conventional PDMM for a specific network topology and updating order. FS-PDMM, as presented in Algorithm 1, requires only the equivalent of a *single pass* over the iterates (4.3a)-(4.3c), while retaining the same convergence properties as PDMM as presented in [158]. We summarize this in the following theorem:

Theorem 1. *Let FS-PDMM (Algorithm 1) be applied over a node set $\mathcal{V}$ with edge set $\mathcal{E}$. This is equivalent to performing PDMM over the same network, but where each node i now has an additional node $i+V$ attached only to it. The node i holds the cost function f_i while the node $i+V$ holds the regularization function R_i, for all nodes $i \in \mathcal{V}$. FS-PDMM therefore converges at rate $O(1/k)$ when using cyclic updates.*

Proof. We begin by restating the FS-PDMM update iterations from Algorithm 1 using the definition of the proximal operator (4.4). At each iteration k we select a random node i and therefore perform

$$\boldsymbol{x}_i^{k+1} = \arg\min_{\boldsymbol{x}_i} \Bigg[f_i(\boldsymbol{x}_i) + \boldsymbol{x}_i^T \Big(\sum_{j\in\mathcal{V}_i} \boldsymbol{A}_{i|j}^T \boldsymbol{\lambda}_{j|i}^k + \boldsymbol{M}_i^T \boldsymbol{\nu}_i^k \Big) + \sum_{j\in\mathcal{V}_i} \frac{\rho}{2} \|\boldsymbol{A}_{i|j}\boldsymbol{x}_i - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^k\|_2^2 + \frac{\rho}{2}\|\boldsymbol{M}_i\boldsymbol{x}_i - \boldsymbol{z}_i^k\|_2^2 \Bigg] \tag{4.5a}$$

$$\boldsymbol{\lambda}_{i|j}^{k+1} = \rho(\boldsymbol{A}_{j|i}\boldsymbol{x}_j^k - \boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1}) - \boldsymbol{\lambda}_{j|i}^k \quad \forall j \in \mathcal{V}_i \tag{4.5b}$$

$$\boldsymbol{\eta}_i^{k+1} = \rho(\boldsymbol{z}_i^k - \boldsymbol{M}_i\boldsymbol{x}_i^{k+1}) - \boldsymbol{\nu}_i^k \tag{4.5c}$$

$$\boldsymbol{z}_i^{k+1} = \arg\min_{\boldsymbol{z}_i} \left[\frac{\rho}{2} \|\boldsymbol{z}_i - (\boldsymbol{M}_i \boldsymbol{x}_i^{k+1} - \boldsymbol{\eta}_i^{k+1}/\rho)\|_2^2 + R_i(\boldsymbol{z}_i) \right] \tag{4.5d}$$

$$\boldsymbol{\nu}_i^{k+1} = \rho(\boldsymbol{M}_i \boldsymbol{x}_i^{k+1} - \boldsymbol{z}_i^{k+1}) - \boldsymbol{\eta}_i^{k+1} \tag{4.5e}$$

$$\begin{aligned} &(\boldsymbol{x}_m^{k+1}, \boldsymbol{z}_m^{k+1}, \boldsymbol{\nu}_m^{k+1}, \boldsymbol{\eta}_i^{k+1}, \boldsymbol{\lambda}_{m|q}^{k+1}) \\ &\quad = (\boldsymbol{x}_m^k, \boldsymbol{z}_m^k, \boldsymbol{\nu}_m^k, \boldsymbol{\eta}_i^k, \boldsymbol{\lambda}_{m|q}^k) \forall m \in \mathcal{V}, m \neq i, \forall q \in \mathcal{V}_m, \end{aligned} \tag{4.5f}$$

where we have introduced the extra intermediate variable $\boldsymbol{\eta}_i^{k+1}$ rather than including $\boldsymbol{z}_i^k$ explicitly in the minimization over $\boldsymbol{z}_i$, for notational convenience. Next we introduce a "virtual" node index $t = i + V$ for each $i \in \mathcal{V}$ and collect these new indices in the set $\mathcal{W} = \{V+1, \ldots, V+V\}$. We then create a larger index set $\mathcal{U} = \mathcal{V} \cup \mathcal{W}$, which contains all physical node indices and the extra "virtual" indices. As with our physical neighbourhood sets $\mathcal{V}_i$, we may now define neighbourhood sets of this larger index set as

$$\mathcal{U}_i = \begin{cases} \mathcal{V}_i \cup \{i+V\} & \text{for } i \in \mathcal{V} \\ \\ \{i, i-V\} & \text{for } i \in \mathcal{W}. \end{cases} \tag{4.6}$$

These new neighbourhood sets $\mathcal{U}_i$ represent the original structure of our physical network, with the addition of an extra "virtual" node $t = i + V$ attached to each node i. Relabelling our optimization variables $\boldsymbol{z}_i^k = \boldsymbol{x}_t^k$, $\boldsymbol{\nu}_i^k = \boldsymbol{\lambda}_{t|i}^k$, and $\boldsymbol{\eta}_i^k = \boldsymbol{\lambda}_{i|t}^k$ then allows the update iterates (4.5a)-(4.5f) for a triggered node i to be expressed as

$$\begin{aligned} \boldsymbol{x}_i^{k+1} = \arg\min_{\boldsymbol{x}_i} \Bigg[& f_i(\boldsymbol{x}_i) + \boldsymbol{x}_i^T \Big(\sum_{j \in \mathcal{U}_i} \boldsymbol{B}_{i|j}^T \boldsymbol{\lambda}_{j|i}^k \Big) \\ & + \sum_{j \in \mathcal{U}_i} \frac{\rho}{2} \|\boldsymbol{B}_{i|j} \boldsymbol{x}_i - \boldsymbol{B}_{j|i} \boldsymbol{x}_j^k\|_2^2 \Bigg] \end{aligned} \tag{4.7a}$$

$$\boldsymbol{\lambda}_{i|j}^{k+1} = \rho(\boldsymbol{B}_{j|i} \boldsymbol{x}_j^k - \boldsymbol{B}_{i|j} \boldsymbol{x}_i^{k+1}) - \boldsymbol{\lambda}_{j|i}^k \quad \forall j \in \mathcal{U}_i \tag{4.7b}$$

$$\begin{aligned} \boldsymbol{x}_t^{k+1} = \arg\min_{\boldsymbol{x}_t} \Bigg[& R_t(\boldsymbol{x}_t) + \boldsymbol{x}_t^T \boldsymbol{B}_{t|i}^T \boldsymbol{\lambda}_{i|t}^{k+1} \\ & + \frac{\rho}{2} \|\boldsymbol{B}_{t|i} \boldsymbol{x}_t - \boldsymbol{B}_{i|t} \boldsymbol{x}_i^{k+1}\|_2^2 \Bigg] \quad t = i + V \end{aligned} \tag{4.7c}$$

$$\boldsymbol{\lambda}_{t|i}^{k+1} = \rho(\boldsymbol{B}_{i|t}\boldsymbol{x}_i^{k+1} - \boldsymbol{B}_{t|i}\boldsymbol{x}_t^{k+1}) - \boldsymbol{\lambda}_{i|t}^{k+1} \quad t = i + V \tag{4.7d}$$

$$(\boldsymbol{x}_m^{k+1},\boldsymbol{\lambda}_{m|q}^{k+1}) = (\boldsymbol{x}_m^k, \boldsymbol{\lambda}_{m|q}^k)\ \forall m \in \mathcal{U}, m \neq i \vee t, \forall q \in \mathcal{U}_m, \tag{4.7e}$$

where the new constraint matrices $(\boldsymbol{B}_{i|j}, \boldsymbol{B}_{j|i})$ are the same as our original matrices $(\boldsymbol{A}_{i|j}, \boldsymbol{A}_{j|i})$ with the addition of two new constraints for each extra "virtual" node, i.e.,

$$\boldsymbol{B}_{i|j} = \boldsymbol{A}_{i|j} \quad \forall (i,j) \in \mathcal{E}, \tag{4.8}$$

$$\boldsymbol{B}_{i|t} = \boldsymbol{M}_i \quad \text{for } t = i + V, \tag{4.9}$$

$$\boldsymbol{B}_{t|i} = \boldsymbol{I}_i \quad \text{for } t = i + V, \tag{4.10}$$

where $\boldsymbol{M}_i$ is defined in our initial problem formulation (4.1), and we have changed the index of our regularization function R_t to reflect the change of variable index. Note that all of these changes are simple relabellings and do not change the optimization problem or the values of variables computed at each iteration.

The update iterations (4.7a)-(4.7e) are identical to those performed for two successive iterations of PDMM as presented in [156] over a network described by the node set $\mathcal{U}$, the edge set $(\mathcal{E} \cup \{(t,i)|t = i + V\})$, with edge constraints $(\boldsymbol{B}_{i|j}, \boldsymbol{B}_{j|i})$ as defined in (6.5)-(6.7), where nodes i and $t = i+V$ hold the cost function f_i and regularization function R_i, respectively, of our original problem (4.1). ■

We have essentially embedded our problem in a "virtual" network with $2V$ nodes, where the topology is the same as our physical network with the addition of an extra "virtual" node attached to each physical node i. Our cost functions f_i and regularization functions R_i are split between these physical and "virtual" node pairs, and we trigger the pair successively when triggering a physical node i.

4.3 Numerical Simulation

In this section we present the results of simulations using FS-PDMM to perform l_1 data fitting and elastic net regularized least-squares (ENLS) optimization over a randomly connected graph. The local data and related decision variable $\boldsymbol{x}_i$ are kept locally at node i for optimization. Neighbouring nodes $l \in \mathcal{V}_i$ may share some elements with node i while using their own local data for collaborative optimization.

4.3.1 l_1 Data Fitting

We begin with the l_1 data fitting consensus problem described as

$$\begin{aligned} &\text{minimize} \quad \sum_{j \in \mathcal{V}_i} \|\boldsymbol{x}_i - \boldsymbol{a}_i\|_1 \\ &\text{subject to} \quad \boldsymbol{x}_i = \boldsymbol{x}_j \quad \forall (i,j) \in \mathcal{E}, \end{aligned} \tag{4.11}$$

where $\boldsymbol{a}_i$ are data vectors held by each node $i \in \mathcal{V}$. The optimization goal is to find the common vector $\boldsymbol{x}$ that best fits all distributed data vectors in the l_1 sense. As demonstrated in [127] there are some instances where PDMM directly applied to problem (4.11) fails to converge and instead oscillates indefinitely around a suboptimal point. The reasons for this lack of convergence require knowledge of monotone operator theory and are beyond the scope of this study. We will, however, show experimentally that performing function splitting results in stable convergence for optimization when compared to direct PDMM optimization.

To apply FS-PDMM to problem (4.11) we define our cost function as $f_i(\boldsymbol{x}_i) = 0$ and our regularization function as $R_i(\boldsymbol{x}_i) = \|\boldsymbol{x}_i - \boldsymbol{a}_i\|_1$ giving us

$$\begin{aligned} &\text{minimize} \quad \sum_{j \in \mathcal{V}_i} f_i(\boldsymbol{x}_i) + R_i(\boldsymbol{x}_i) \\ &\text{subject to} \quad \boldsymbol{x}_i = \boldsymbol{x}_j \quad \forall (i,j) \in \mathcal{E}, \end{aligned} \tag{4.12}$$

where $\boldsymbol{M}_i = I \ \forall i \in \mathcal{V}$ and $\boldsymbol{A}_{i|j} = \boldsymbol{A}_{j|i} = I \ \forall (i,j) \in \mathcal{E}$. Compared with the FS-PDMM algorithm, we note that the extra constant term $\boldsymbol{a}_i$ in the

regularization function R requires a minor adjustment to the FS-PDMM update equations, but the general algorithm remains the same. We therefore have the following FS-PDMM update iterations for the l_1 data fitting consensus problem problem

$$\boldsymbol{x}_i^{k+1} = \frac{1}{2\rho V_i}\left(\rho \boldsymbol{z}_i^k + \rho \boldsymbol{a}_i - \boldsymbol{\nu}_i^k - \sum_{j\in\mathcal{V}_i}(\boldsymbol{\lambda}_{ji}^k + \rho \boldsymbol{x}_j^k)\right) \tag{4.13}$$

$$\boldsymbol{z}_i^{k+1} = \mathbf{prox}_{l_i,1/\rho}(2\boldsymbol{x}_i^{k+1} - 2\boldsymbol{a}_i - \boldsymbol{z}_i^k + \boldsymbol{\nu}_i^k/\rho) \tag{4.14}$$

$$\boldsymbol{\nu}_i^{k+1} = \rho(2\boldsymbol{x}_i^{k+1} - 2\boldsymbol{a}_i - \boldsymbol{z}_i^{k+1} - \boldsymbol{z}_i^k) + \boldsymbol{\nu}_i^k \tag{4.15}$$

$$\boldsymbol{\lambda}_{i|j}^{k+1} = \rho(\boldsymbol{x}_j^k - \boldsymbol{x}_i^{k+1}) - \boldsymbol{\lambda}_{j|i}^k \qquad \forall j \in \mathcal{V}_i, \tag{4.16}$$

for all $i \in \mathcal{V}$. Recalling the equivalence analysis from 4.2, we have effectively embedded the linear difference $(\boldsymbol{x}_i - \boldsymbol{a}_i)$ at each node in an edge-wise constraint between the node and its "virtual" neighbour.

Since the solution to problem (4.11) is not guaranteed to be unique our error measure will be the mean-square error of the objective function rather than the optimal weight vector, i.e.,

$$\text{MSE}^k = \frac{1}{V}\sum_{i\in\mathcal{V}}(R_i(\boldsymbol{x}_i^k) - R_i(\boldsymbol{x}_i^{\text{opt}}))^2, \tag{4.17}$$

We simulate a randomly connected network of 20 nodes with randomly generated data vectors $\boldsymbol{a}_i$ $\forall i \in \mathcal{V}$ and $\rho = 0.5$ and generate 100 random instances of this setup. Both conventional PDMM and FS-PDMM applied to the l_1 data fitting consensus problem problem and a plot of the function MSE versus update iterations is presented in figure 4.1.

We see that conventional PDMM has a significantly slower convergence rate than FS-PDMM on average. This is due to conventional PDMM failing to converge in approximately one third of random network instances. Conversely, FS-PDMM converges in all instances without getting stuck oscillating around a suboptimal point. The convergence failure in this specific scenario is analysed using monotone operator theory in [127], but it appears that the splitting of cost and regularization functions performed implicitly by FS-PDMM resolves this problem.

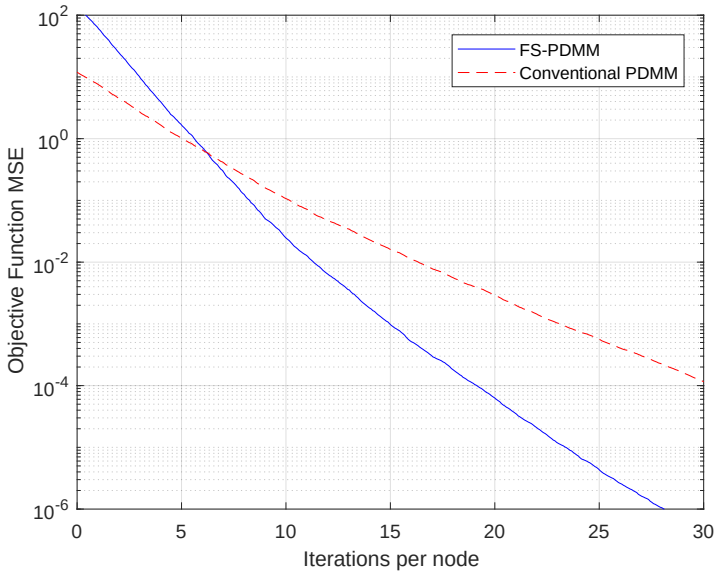


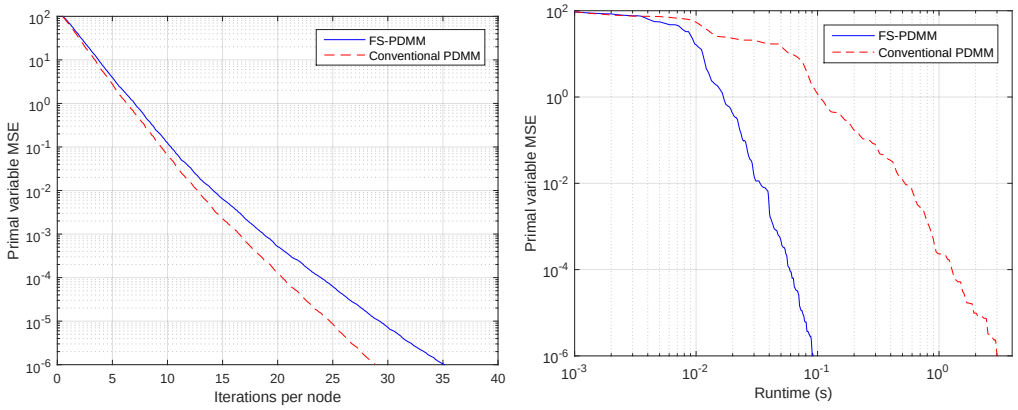
Figure 4.1: Convergence rate of FS-PDMM and conventional PDMM for the l_1 data fitting consensus problem (asynchronous).

4.3.2 Elastic Net Regularized Least-Squares

For ENLS, the local problem function at node i is

$$g_i^{\text{ENLS}}(\boldsymbol{x}_i) = \|\boldsymbol{D}_i\boldsymbol{x}_i - \boldsymbol{c}_i\|^2 + \|\boldsymbol{x}_i\|_1 + \frac{1}{2}\|\boldsymbol{x}_i\|_2^2, \tag{4.18}$$

where $\boldsymbol{D}_i$ and $\boldsymbol{c}_i$ are the data block and error vectors, respectively. This is a canonical sparsity inducing regularization problem involving a simple least squares cost function, where the extra l_2 norm on the optimization variable allows the problem to be strictly convex and therefore have only a single global minima. This means both algorithms will converge on the same optimal point. FS-PDMM performing ENLS was compared to conventional PDMM to determine the convergence rate and the runtime. A randomly connected graph of 10 nodes was formed, each with between five and ten vector elements. Each node randomly shared a subset of these elements with their neighbours to initialise the consensus pattern for the current simulation realisation. At iteration k a node i randomly



(a) Convergence rate of FS-PDMM and conventional PDMM for elastic net regularized least-squares (asynchronous).

(b) Primal variable MSE versus local runtime of FS-PDMM and conventional PDMM for elastic net regularized least-squares (asynchronous).

Figure 4.2: Performance of FS-PDMM and conventional PDMM.

was triggered from a uniform distribution for variable update, followed by a sharing of these updated variables with local neighbours.

The optimal primal variable $\boldsymbol{x}_i^{\text{opt}}$ was calculated for all nodes i to a precision of 10^{-9} prior to distributed optimization. At each iteration of the regularized and conventional PDMM optimization procedure we compute the mean-square error (MSE) across all nodes as

$$\text{MSE}^k = \frac{1}{V} \sum_{i \in \mathcal{V}} \|\boldsymbol{x}_i^k - \boldsymbol{x}_i^{\text{opt}}\|^2, \tag{4.19}$$

and capture the runtime for each triggered node's update routine to complete.

Fig. 4.2a shows the MSE as a function of iterations per node for ENLS using FS-PDMM and conventional PDMM. We see that they both perform similarly, with FS-PDMM requiring around five iterations per node more than conventional PDMM to reach an accuracy of 10^{-6}. However, from Fig. 4.2b we see that FS-PDMM has a runtime over an order of magnitude less than conventional PDMM, which is performing local variable splitting

and minimization using ADMM as in equations (4.3a)-(4.3c).

We would like to reiterate that the FS-PDMM algorithm, while providing lower computational complexity and runtimes, also does not require any additional numerical optimizers. Practically, this allows our algorithm to be implemented in systems where processing units have low memory and are not able to store libraries of solvers. Large wireless sensor networks, for example, often have memory and processor restrictions at each node that may make this overhead requirement infeasible.

4.4 Summary

In this section we proposed a distributed algorithm based on PDMM called FS-PDMM for local function splitting with the aim of reducing local subproblem computation times. We showed that the FS-PDMM algorithm is equivalent to conventional PDMM performed over an altered network toplogy, and therefore retains the O(1/k) convergence rate for conventional PDMM. Simulated experiments show that FS-PDMM performs local update iterations roughly an order of magnitude faster than conventional PDMM without the need for additional numerical optimizers, while increasing the number of iterations until convergence slightly.

Chapter 5

Quadratic Approximation PDMM

Given the computational savings as well as the simple form of many local updates resulting from the function splitting in chapter 4, a natural question to ask is whether the update efficiency of individual functions may be improved. In this chapter, inspired by works such as [92] and [23], we show that when our local cost functions $f_i(\boldsymbol{x}_i)$ are twice differentiable with Lipschitz continuous gradients it is possible to quadratically approximate local updates of these functions using the gradient information of these functions. This greatly reduces the local computational cost of our distributed optimization when utilizing certain cost functions, while still allowing for rapid network convergence. Additionally, as in the case of function splitting, this also often eliminates the need for complex optimization packages that increasee the overhead of distributed sensors. We will refer to the algorithm resulting from this approximation as Quadratically Approximated PDMM (QA-PDMM).

We begin by stating the quadratic approximation made to the local cost function at each node and present the associated primal variable update equations. We then present two convergence analyses for synchronous QA-PDMM, where the first assumes Lipschitz continuity and strong convexity of local functions and the second removes the assumption of strong convexity. We provide sufficient conditions for convergence and show that

these lead to a guaranteed sublinear convergence rate of $O(1/k)$. Finally, we present simulated experiments that confirm the effectiveness of QA-PDMM when applied to a distributed ridge regularized logistic regression problem over a general random graph with partial consensus.

5.1 The QA-PDMM Algorithm

Inspired by the computational savings demonstrated in [92], in this section we develop a PDMM updating scheme that uses a quadratic approximation of the local cost function f_i when performing local primal variable updates. Recall from section 2.2.5 that PDMM optimizes the decomposable problem form (2.32) using the update equations

$$\begin{aligned}\boldsymbol{x}_i^{k+1} = \arg\min_{\boldsymbol{x}_i} \Big[f_i(\boldsymbol{x}_i) + \sum_{j\in\mathcal{V}_i} \boldsymbol{\lambda}_{j|i}^{k,T} \boldsymbol{A}_{i|j}\boldsymbol{x}_i \\ + \frac{1}{2}\|\boldsymbol{A}_{i|j}\boldsymbol{x}_i - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^k\|^2_{\boldsymbol{P}_{ij}} \Big] \quad i\in\mathcal{V}\end{aligned} \tag{5.1}$$

$$\boldsymbol{\lambda}_{i|j}^{k+1} = \boldsymbol{P}_{ij}(\boldsymbol{A}_{j|i}\boldsymbol{x}_j^k - \boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1}) - \boldsymbol{\lambda}_{j|i}^k \quad i\in\mathcal{V}, j\in\mathcal{V}_i \tag{5.2}$$

where each $\boldsymbol{P}_{ij}$ is a positive definite matrix (i.e., $\boldsymbol{P}_{ij} \succ 0$), and $\|\cdot\|_P$ represents the weighted Euclidean norm by the matrix P.

For some objective functions $\{f_i | i\in\mathcal{V}\}$, such as the softmax function in logistic regression, it might be expensive to compute the exact solution $\{\boldsymbol{x}_i^{k+1} | i\in\mathcal{V}\}$ in (5.1). In those situations, it would be computationally beneficial to approximately perform the primal variable update if convergence was still guaranteed. For this section we make the additional assumption that each convex cost function $f_i : \mathbb{R}^{n_i} \to \mathbb{R}$ is continuously differentiable with the Lipschitz continuous gradient $Q_i > 0$:

$$\|\nabla f_i(\boldsymbol{x}_i) - \nabla f_i(\boldsymbol{y}_i)\| \le Q_i\|\boldsymbol{x}_i - \boldsymbol{y}_i\| \quad \forall \boldsymbol{x}_i, \boldsymbol{y}_i \in \mathbb{R}^{n_i}, \tag{5.3}$$

where $\|\cdot\|$ denotes the standard Euclidean norm. This implies that there is an upper bound on how rapidly the derivative of the function f_i can

change. Given this assumption, we quadratically approximate the cost function f_i at $\boldsymbol{x}_i^k$ as

$$f_i^k(\boldsymbol{x}_i) = f_i(\boldsymbol{x}_i^k) + (\boldsymbol{x}_i - \boldsymbol{x}_i^k)^T \nabla f_i(\boldsymbol{x}_i^k) + \frac{1}{2}\|\boldsymbol{x}_i - \boldsymbol{x}_i^k\|_{H_i^k}^2 \quad i \in \mathcal{V}, \tag{5.4}$$

where $H_i^k \succ 0$ is a positive-definite weighting matrix for node i at iteration k. This approximation is then used in the primal variable update (5.1), giving the new update equations

$$\boldsymbol{x}_i^{k+1} = \arg\min_{\boldsymbol{x}_i} \Big[f_i(\boldsymbol{x}_i^k) + (\boldsymbol{x}_i - \boldsymbol{x}_i^k)^T \nabla f_i(\boldsymbol{x}_i^k) + \frac{1}{2}\|\boldsymbol{x}_i - \boldsymbol{x}_i^k\|_{H_i^k}^2 + \boldsymbol{x}_i^T \Big(\sum_{j \in \mathcal{V}_i} \boldsymbol{A}_{i|j}^T \boldsymbol{\lambda}_{j|i}^k \Big) + \sum_{j \in \mathcal{V}_i} \frac{1}{2}\|\boldsymbol{A}_{i|j}\boldsymbol{x}_i - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^k\|_{\boldsymbol{P}_{ij}}^2 \Big] \quad \forall i \in \mathcal{V} \tag{5.5}$$

$$\boldsymbol{\lambda}_{i|j}^{k+1} = \boldsymbol{P}_{ij}(\boldsymbol{A}_{j|i}\boldsymbol{x}_j^k - \boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1}) - \boldsymbol{\lambda}_{j|i}^k \qquad \forall j \in \mathcal{V}_i,\ \forall i \in \mathcal{V}. \tag{5.6}$$

QA-PDMM therefore attempts to simplify the optimization (5.1) by using the gradient information of the objective function computed at the most recent estimate, rather than performing an exact minimization. The only condition required for convergence of QA-PDMM is that the weighting matrix minus the Lipschitz constant multiplied by the identity matrix be positive definite, i.e. $H_i^k - Q_i I_i \succ 0$.

Simplifying this primal update to give an explicit expression yields QA-PDMM presented in Algorithm 2, the convergence of which will be analyzed in Section 5.2. The update procedure for QA-PDMM requires only a single local matrix inversion and linear transformations for the primal variable update, and only simple linear transformations for the dual update. This allows for large computational savings for certain optimization problems, such as logistic regression, as we will see in Section 5.3.

Note that while the general PDMM algorithm may operate synchronously or asynchronously across a network, we have restricted QA-PDMM to synchronous updates. This is due to the convergence analysis performed in Subsection 5.2, which assumes synchronous operation.

Algorithm 2 QA-PDMM

Initialize:

specify $f_i, \boldsymbol{A}_{i|j}, \rho, H_i^k, \epsilon \qquad \forall i \in \mathcal{V}, \forall j \in \mathcal{V}_i$

randomly initialize $(\boldsymbol{x}_i, \boldsymbol{\lambda}_{i|j}) \qquad \forall i \in \mathcal{V}, \forall j \in \mathcal{V}_i$

Update procedure:

while *stopping criterion* $==$ *false* **do**

at each iteration k, all nodes i synchronously perform

$$\boldsymbol{x}_i^{k+1} = (\textstyle\sum_{j\in\mathcal{V}_i} \boldsymbol{A}_{i|j}^T \boldsymbol{P}_{ij} \boldsymbol{A}_{i|j} + H_i^k)^{-1} \Big[H_i^k \boldsymbol{x}_i^k - \nabla f_i(\boldsymbol{x}_i^k) + \textstyle\sum_{j\in\mathcal{V}_i} \boldsymbol{A}_{i|j}^T (\boldsymbol{P}_{ij} \boldsymbol{A}_{j|i} \boldsymbol{x}_j^k - \boldsymbol{\lambda}_{j|i}^k) \Big]$$

$$\boldsymbol{\lambda}_{i|j}^{k+1} = \boldsymbol{P}_{ij}(\boldsymbol{A}_{j|i}\boldsymbol{x}_j^k - \boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1}) - \boldsymbol{\lambda}_{j|i}^k \qquad \forall j \in \mathcal{V}_i$$

share $\boldsymbol{x}_i^{k+1}, \boldsymbol{\lambda}_{i|j}^{k+1}$ with neighbours $j \in \mathcal{V}_i$

if $\|\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k\| + 1/V_i \textstyle\sum_{j\in\mathcal{V}_i} \|\boldsymbol{\lambda}_{i|j}^{k+1} - \boldsymbol{\lambda}_{i|j}^k\| < \epsilon$ **then**

stopping criterion $=$ *true*

end if

$k \leftarrow k + 1$

end while

5.2 QA-PDMM Convergence

In this section we will present two convergence analyses that demonstrate the convergence properties of QA-PDMM. Synchronous PDMM converges to an optimal solution $\lim_{k\to\infty}(\boldsymbol{x}^k, \boldsymbol{\lambda}^k) = (\boldsymbol{x}^\star, \boldsymbol{\lambda}^\star)$, where $\boldsymbol{x} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{|\mathcal{V}|}]^T$ and $\boldsymbol{\lambda} = [\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots, \boldsymbol{\lambda}_{|\mathcal{V}|}]^T$, for the problem (2.32) if and only if $(\boldsymbol{x}^\star, \boldsymbol{\lambda}^\star)$ satisfies the following optimality conditions [158, 100]

$$\nabla f_i(\boldsymbol{x}_i^\star) = \sum_{j\in\mathcal{V}_i} \boldsymbol{A}_{i|j}^T \boldsymbol{\lambda}_{i|j}^\star \qquad i \in \mathcal{V} \tag{5.7}$$

$$\boldsymbol{\lambda}_{i|j}^\star = \boldsymbol{\lambda}_{j|i}^\star \qquad (i,j) \in \mathcal{E} \tag{5.8}$$

$$\boldsymbol{A}_{i|j}\boldsymbol{x}_i^\star = \boldsymbol{A}_{j|i}\boldsymbol{x}_j^\star \qquad (i,j) \in \mathcal{E}. \tag{5.9}$$

We will show in next section that if the set of parameters $\{H_i | i \in \mathcal{V}\}$ are properly chosen in the approximation (5.4), then synchronous QA-PDMM converges to an optimal point.

5.2.1 Convergence Analysis 1

In our first analysis we assume that the local objective functions $f_i(\boldsymbol{x}_i)$ are twice differentiable and the eigenvalues of their local Hessians $\nabla^2 f_i(\boldsymbol{x}_i)$ are bounded by positive constants q_i and Q_i where $0 < q_i \leq Q_i < \infty \forall i \in \mathcal{V}$. Therefore, for any $\boldsymbol{x}_i \in \mathbb{R}^{n_i}$ we have

$$q_i I_i \preceq \nabla^2 f_i(\boldsymbol{x}_i) \preceq Q_i I_i \quad \forall i \in \mathcal{V}, \tag{5.10}$$

where $I_i \in \mathbb{R}^{n_i \times n_i}$ is the identity matrix. The lower bound on the Hessian implies that for all $i \in \mathcal{V}$, each local function f_i is strongly convex with constant q_i, i.e.,

$$(\nabla f_i(\boldsymbol{x}_i) - \nabla f_i(\boldsymbol{y}_i))^T(\boldsymbol{x}_i - \boldsymbol{y}_i) \geq q_i \|\boldsymbol{x}_i - \boldsymbol{y}_i\|_2^2 \, \forall \boldsymbol{x}_i, \boldsymbol{y}_i \in \mathbb{R}^{n_i}, \tag{5.11}$$

while the upper bound is equivalent to Lipschitz continuous gradients of the local functions f_i for all $i \in \mathcal{V}$ with constant Q_i, i.e.,

$$\|\nabla f_i(\boldsymbol{x}_i) - \nabla f_i(\boldsymbol{y}_i)\|_2 \leq Q_i \|\boldsymbol{x}_i - \boldsymbol{y}_i\|_2 \, \forall \boldsymbol{x}_i, \boldsymbol{y}_i \in \mathbb{R}^{n_i}. \tag{5.12}$$

Additionally, we assume that the tuning matrices $\boldsymbol{P}_{ij} = \rho \ \forall (i,j) \in \mathcal{E}$ for some scalar constant $\rho > 0$.

We begin by forming an inequality that captures the convergence of our primal and dual variables.

Lemma 1. *Given the assumption on the Hessian of each cost function f_i as stated in (5.10), for some scalar $c_i > 0 \ \forall i \in \mathcal{V}$ we may form the inequality*

$$\begin{aligned}
&\sum_{i\in\mathcal{V}} \Bigg[(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^*)^T \Big[H_i^k + (2q_i - c_i) I_i \Big] (\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^*) \\
&\quad + (\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k)^T \Big[H_i^k - \frac{Q_i}{c_i} I_i \Big] (\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k) \\
&+ \frac{1}{2} \sum_{j\in\mathcal{V}_i} \Big[\|\rho^{-1/2}(\boldsymbol{\lambda}_{i|j}^* + \boldsymbol{\lambda}_{j|i}^{k+1}) + \rho^{1/2}(\boldsymbol{A}_{i|j}\boldsymbol{x}_i^* - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^{k+1})\|^2 \\
&\quad + \|\rho^{-1/2}(\boldsymbol{\lambda}_{i|j}^{k+1} + \boldsymbol{\lambda}_{j|i}^{k}) + \rho^{1/2}(\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1} - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^{k})\|^2 \Big] \Bigg] \\
&\leq \sum_{i\in\mathcal{V}} \|\boldsymbol{x}_i^k - \boldsymbol{x}_i^*\|_{H_i^k}^2 \\
&\quad + \frac{1}{2} \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{V}_i} \|\rho^{-1/2}(\boldsymbol{\lambda}_{i|j}^* + \boldsymbol{\lambda}_{j|i}^k) + \rho^{1/2}(\boldsymbol{A}_{i|j}\boldsymbol{x}_i^* - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^k)\|^2. \qquad (5.13)
\end{aligned}$$

Proof. See the proof in Appendix A.1. ■

Using the inequality of Lemma 1, we may present the convergence guarantees for QA-PDMM in the following theorem.

Theorem 2. *If each weighting matrix H_i^k is selected such that*

$$H_i^k - \frac{Q_i}{2q_i} I_i \succ 0 \qquad (5.14)$$

we have

$$\lim_{k\to\infty} \boldsymbol{x}_i^k - \boldsymbol{x}_i^* = 0 \qquad \forall i \in \mathcal{V} \qquad (5.15)$$

$$\lim_{k\to\infty} \boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k = 0 \qquad \forall i \in \mathcal{V}, \qquad (5.16)$$

and

$$\lim_{k\to\infty} \boldsymbol{A}_{i|j}\boldsymbol{x}_i^k = \boldsymbol{A}_{j|i}\boldsymbol{x}_j^k \qquad \forall (i,j) \in \mathcal{E} \tag{5.17}$$

$$\lim_{k\to\infty} \boldsymbol{\lambda}_{i|j}^k + \boldsymbol{\lambda}_{j|i}^k = 0 \qquad \forall (i,j) \in \mathcal{E}, \tag{5.18}$$

i.e., each primal variable estimate $\boldsymbol{x}_i^k$ converges to the optimal value $\boldsymbol{x}_i^*$ $\forall i \in \mathcal{V}$, the distance between estimates $\boldsymbol{x}_i^{k+1}$ and $\boldsymbol{x}_i^k$ tends to zero, and our estimate converges to a primal and dual feasible point.

Proof. Let c_i be a positive real number such that $2q_i - c_i > 0$ for each $i \in \mathcal{V}$. Then $H_i^k - (Q_i/c_i)I_i \succeq 0$ and $H_i^k + (2q_i - c_i)I_i \succeq 0$, i.e., the first two quadratic terms in (5.13) are positive. This leads to (5.15) and (5.16).

Secondly, Lemma 1 implies that

$$\begin{aligned}
&\lim_{k\to\infty} \Big[\rho^{-1/2}(\boldsymbol{\lambda}_{i|j}^* + \boldsymbol{\lambda}_{j|i}^k) \\
&\qquad + \rho^{1/2}(\boldsymbol{A}_{i|j}\boldsymbol{x}_i^* - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^k)\Big] = 0 \quad \forall i \in \mathcal{V}, \forall j \in \mathcal{V}_i \\
&\lim_{k\to\infty} \Big[\rho^{-1/2}(\boldsymbol{\lambda}_{i|j}^{k+1} + \boldsymbol{\lambda}_{j|i}^k) \\
&\qquad + \rho^{1/2}(\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1} - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^k)\Big] = 0 \quad \forall i \in \mathcal{V}, \forall j \in \mathcal{V}_i,
\end{aligned}$$

which results in

$$\begin{aligned}
&\lim_{k\to\infty} \Big[\rho^{-1/2}(\boldsymbol{\lambda}_{i|j}^{k+1} + \boldsymbol{\lambda}_{j|i}^{k+1}) \\
&\qquad + \rho^{1/2}(\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1} - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^{k+1})\Big] = 0 \quad \forall i \in \mathcal{V}, \forall j \in \mathcal{V}_i.
\end{aligned}$$

Therefore, for each edge $(i,j) \in \mathcal{E}$ we may write

$$\begin{aligned}
&\lim_{k\to\infty} \Big[\rho^{-1/2}(\boldsymbol{\lambda}_{i|j}^k + \boldsymbol{\lambda}_{j|i}^k) + \rho^{1/2}(\boldsymbol{A}_{i|j}\boldsymbol{x}_i^k - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^k)\Big] = 0 \\
&\lim_{k\to\infty} \Big[\rho^{-1/2}(\boldsymbol{\lambda}_{i|j}^k + \boldsymbol{\lambda}_{j|i}^k) + \rho^{1/2}(\boldsymbol{A}_{j|i}\boldsymbol{x}_j^k - \boldsymbol{A}_{i|j}\boldsymbol{x}_i^k)\Big] = 0,
\end{aligned}$$

leading to (5.17) and (5.18). ■

5.2.2 Convergence Analysis 2

In this section we will show that the lower bound on the Hessian eigenvalues (5.10), which imply strong convexity of the functions f_i, is not necessary to guarantee convergence of QA-PDMM. This second study makes use of the analysis approach in [6] developed for the Fast Iterative-Shrinkage Thresholding Algorithm (FISTA), and was lead by Dr. G. Zhang. We first construct a special inequality for each $\boldsymbol{x}_i^{k+1}$ in (5.5) and then exploit this to analyze synchronous QA-PDMM.

We begin by introducing a standard inequality for each f_i with Lipschitz continuous gradient Q_i in (6.2). Let each f_i in (2.32) be a continuously differentiable function with the Lipschitz continuous gradient Q_i, then for any $h_i \geq Q_i$ (Lemma 2.3 in [6])

$$f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{y}) + (\boldsymbol{x}-\boldsymbol{y})^T \nabla f_i(\boldsymbol{y}) + \frac{h_i}{2}\|\boldsymbol{x}-\boldsymbol{y}\|^2 \ \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n. \tag{5.19}$$

Next, from (5.5)-(5.6) the optimality condition for each $\boldsymbol{x}_i^{k+1}$ can be easily derived as

$$\nabla f_i(\boldsymbol{x}_i^k) + Q_i(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k) = \sum_{j \in \mathcal{V}_i} A_{i|j}^T \boldsymbol{\lambda}_{i|j}^{k+1} \quad i \in \mathcal{V}. \tag{5.20}$$

With (5.19) and (5.20) we may now derive an inequality for each $\boldsymbol{x}_i^{k+1}$ in (5.5), summarized in the following Lemma:

Lemma 2. *Let $h_i \geq Q_i$ in the approximation function (5.4). Then for any $\boldsymbol{x}_i \in \mathbb{R}^{n_i}$,*

$$\begin{aligned} f_i(\boldsymbol{x}_i) - f_i(\boldsymbol{x}_i^{k+1}) \geq \frac{h_i}{2}\|\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k\|^2 - h_i(\boldsymbol{x}_i - \boldsymbol{x}_i^k)^T(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k) \\ + (\boldsymbol{x}_i - \boldsymbol{x}_i^{k+1})^T \sum_{j \in \mathcal{V}_i} A_{i|j}^T \boldsymbol{\lambda}_{i|j}^{k+1}. \end{aligned} \tag{5.21}$$

Proof. From (5.4) and (5.19), we have

$$f_i(\boldsymbol{x}_i) - f_i(\boldsymbol{x}_i^{k+1})$$

$$
\begin{aligned}
&\geq f_i(\boldsymbol{x}_i) - f_i^k(\boldsymbol{x}_i^{k+1}) \\
&\overset{(a)}{\geq} f_i(\boldsymbol{x}_i^k) + (\boldsymbol{x}_i - \boldsymbol{x}_i^k)^T \nabla f_i(\boldsymbol{x}_i^k) - f_i^k(\boldsymbol{x}_i^{k+1}) \\
&= f_i(\boldsymbol{x}_i^k) + (\boldsymbol{x}_i - \boldsymbol{x}_i^k)^T \nabla f_i(\boldsymbol{x}_i^k) \\
&\quad - \left[f_i(\boldsymbol{x}_i^k) + (\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k)^T \nabla f_i(\boldsymbol{x}_i^k) + \frac{h_i}{2} \|\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k\|^2 \right] \\
&= (\boldsymbol{x}_i - \boldsymbol{x}_i^{k+1})^T \nabla f_i(\boldsymbol{x}_i^k) - \frac{h_i}{2} \|\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k\|^2 \\
&\overset{(b)}{=} (\boldsymbol{x}_i - \boldsymbol{x}_i^{k+1})^T \left(\sum_{j \in \mathcal{V}_i} A_{i|j}^T \boldsymbol{\lambda}_{i|j}^{k+1} - h_i(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k) \right) \\
&\quad - \frac{h_i}{2} \|\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k\|^2,
\end{aligned}
$$

where step (a) uses the property that f_i is a convex function and step (b) uses (5.20). The above expression can then be further simplified as (5.21) using algebra. ■

We may now derive the convergence properties of synchronous QA-PDMM based on Lemma 2. The derivation procedure is similar to the work in [157] for analyzing synchronous PDMM. Suppose $(\boldsymbol{x}^\star, \boldsymbol{\lambda}^\star)$ is an optimal point satisfying (5.7)-(5.9). We first derive an upper and lower bound for the quantity $\sum_{i \in \mathcal{V}} \left[f_i(\boldsymbol{x}_i^{k+1}) - f_i(\boldsymbol{x}_i^\star) - \boldsymbol{x}_i^{k+1,T} \sum_{j \in \mathcal{V}_i} A_{i|j}^T \boldsymbol{\lambda}_{i|j}^\star \right]$ in a lemma below.

Lemma 3. *Let $(\boldsymbol{x}^\star, \boldsymbol{\lambda}^\star)$ be an optimal solution satisfying (5.7)-(5.9). The estimate $(\boldsymbol{x}^{k+1}, \boldsymbol{\lambda}^{k+1})$ is obtained by performing (5.4)-(5.6) under the condition that $h_i \geq Q_i$, $i \in \mathcal{V}$. Then there is*

$$
\begin{aligned}
0 &\leq 2 \sum_{i \in \mathcal{V}} \left[f_i(\boldsymbol{x}_i^{k+1}) - f_i(\boldsymbol{x}_i^\star) - \boldsymbol{x}_i^{k+1,T} \sum_{j \in \mathcal{V}_i} A_{i|j}^T \boldsymbol{\lambda}_{i|j}^\star \right] \\
&\leq \frac{1}{2} \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}_i} \Big[\|\boldsymbol{P}_{ij}^{-1/2} (\boldsymbol{\lambda}_{i|j}^* + \boldsymbol{\lambda}_{j|i}^k) + \boldsymbol{P}_{ij}^{1/2} (A_{i|j} \boldsymbol{x}_i^* - A_{j|i} \boldsymbol{x}_j^k)\|^2 \\
&\qquad - \|\boldsymbol{P}_{ij}^{-1/2} (\boldsymbol{\lambda}_{i|j}^* + \boldsymbol{\lambda}_{j|i}^{k+1}) + \boldsymbol{P}_{ij}^{1/2} (A_{i|j} \boldsymbol{x}_i^* - A_{j|i} \boldsymbol{x}_j^{k+1})\|^2 \\
&\qquad - \|\boldsymbol{P}_{ij}^{-1/2} (\boldsymbol{\lambda}_{i|j}^{k+1} + \boldsymbol{\lambda}_{j|i}^k) + \boldsymbol{P}_{ij}^{1/2} (A_{i|j} \boldsymbol{x}_i^{k+1} - A_{j|i} \boldsymbol{x}_j^k)\|^2 \Big]
\end{aligned}
$$

$$+\sum_{i\in\mathcal{V}} h_i\|\boldsymbol{x}_i^k-\boldsymbol{x}_i^\star\|^2-\sum_{i\in\mathcal{V}} h_i\|\boldsymbol{x}_i^{k+1}-\boldsymbol{x}_i^\star\|^2 \tag{5.22}$$

where the equality for the lower bound holds if and only if $(\boldsymbol{x}^{k+1},\boldsymbol{\lambda}^{k+1})$ *satisfies*

$$\nabla f_i(\boldsymbol{x}_i^{k+1})=\sum_{j\in\mathcal{V}_i} A_{i|j}^T\boldsymbol{\lambda}_{j|i}^\star \qquad \forall i\in\mathcal{V}. \tag{5.23}$$

Proof. See the proof in Appendix A.2. ■

Next we show that the estimates $(\boldsymbol{x}^{k+1},\boldsymbol{\lambda}^{k+1})$ are always bounded using the results of Lemma 3.

Lemma 4. *Every pair of estimates* $(\hat{\boldsymbol{x}}_i^{k+1},\hat{\boldsymbol{\lambda}}_{i|j}^{k+1})$, $i\in\mathcal{V}$, $j\in\mathcal{V}_i$, $k\geq 0$, *in Lemma 3 is upper bounded by a constant* M *under a squared error criterion:*

$$\left\|\boldsymbol{P}_{ij}^{-\frac{1}{2}}(\boldsymbol{\lambda}_{i|j}^\star+\boldsymbol{\lambda}_{j|i}^{k+1})+\boldsymbol{P}_{ij}^{\frac{1}{2}}(A_{i|j}\boldsymbol{x}_i^\star-A_{j|i}\boldsymbol{x}_j^{k+1})\right\|^2\leq M. \tag{5.24}$$

Proof. One can first prove (5.24) for $k=0$ by using (5.22). The inequality (5.24) for $k>0$ can then be proved recursively. ■

Upon obtaining the results in Lemma 3 and 4, we are ready to present the convergence rate of synchronous GPDMM.

Theorem 3. *Let* $(\boldsymbol{x}^k,\boldsymbol{\lambda}^k)$, $k=1,\ldots,K$, *be obtained by performing (5.4)-(5.6) under the condition that* $h_i\geq Q_i$, $i\in\mathcal{V}$. *The average estimate* $(\bar{\boldsymbol{x}}^K,\bar{\boldsymbol{\lambda}}^K)=(\frac{1}{K}\sum_{k=1}^K\boldsymbol{x}^k,\frac{1}{K}\sum_{k=1}^K\boldsymbol{\lambda}^k)$ *satisfies*

$$0\leq\sum_{i\in\mathcal{V}}\Big[f_i(\bar{\boldsymbol{x}}_i^K)-f_i(\boldsymbol{x}_i^\star)-\bar{\boldsymbol{x}}_i^{K,T}\sum_{j\in\mathcal{V}_i}\boldsymbol{A}_{i|j}^T\boldsymbol{\lambda}_{i|j}^\star\Big]\leq\mathcal{O}\Big(\frac{1}{K}\Big) \tag{5.25}$$

$$\lim_{K\to\infty}\boldsymbol{A}_{i|j}\bar{\boldsymbol{x}}_i^K=\boldsymbol{A}_{j|i}\bar{\boldsymbol{x}}_j^K\quad\forall(i,j)\in\mathcal{E} \tag{5.26}$$

$$\lim_{K\to\infty}\bar{\boldsymbol{\lambda}}_{i|j}^K+\bar{\boldsymbol{\lambda}}_{j|i}^K=\boldsymbol{0}\quad\forall(i,j)\in\mathcal{E}. \tag{5.27}$$

Proof. The proof is similar to that for Theorem 2 in [157]. ■

The conditions $\{h_i \geq Q_i | i \in \mathcal{V}\}$ in Theorem 3 ensures that synchronous QA-PDMM possesses the same convergence rate as synchronous PDMM. Therefore, to guarantee convergence the weighting matrix in (5.5) must be larger than the Lipschitz constant, i.e., $H_i^k \succ Q_i$.

5.3 Numerical Simulation

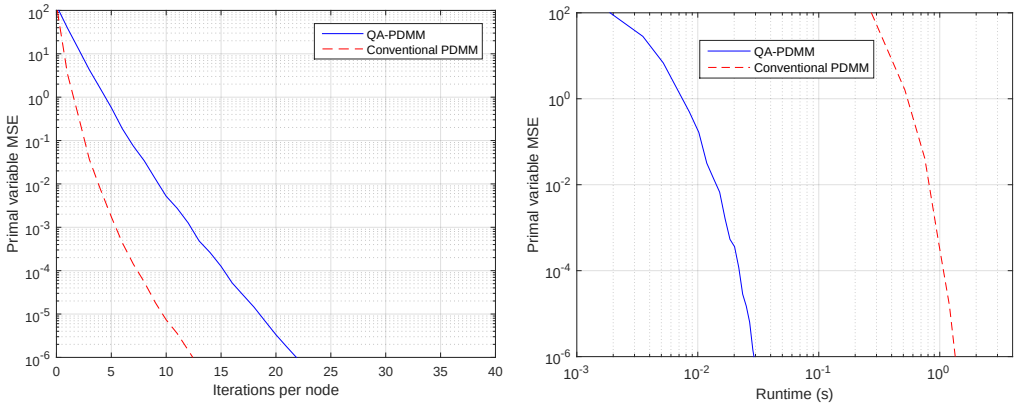
In this section we consider solving the problem of ridge regularized logistic regression (RRLR) over a random graph of 10 nodes using partial consensus constraints. The problem function at each node i is

$$f_i(\boldsymbol{x}_i) = \frac{1}{10}\sum_{p=1}^{10} \log[1 + \exp(-c_{ip}\boldsymbol{d}_{ip}^T \boldsymbol{x}_i)] + \|\boldsymbol{x}_i\|_2^2, \tag{5.28}$$

where each node i holds 10 training points consisting of feature vector $\boldsymbol{d}_{ip} \in \mathbb{R}^{10}$ and binary label c_{ip} for $p = 1, ..., 10$. The logistic term is convex but is computationally expensive to evaluate, where the extra l_2 norm on the optimization variable allows the problem to be strictly convex and therefore have only a single global minima.

The objective is to perform distributed data training in the graph so that after convergence all the nodes reach partial consensus over the random graph. We performed the experiments over a randomly connected graph of 10 nodes, each with between five and ten vector elements. Each node randomly shared a subset of these elements with their neighbours to initialise the consensus pattern for the current simulation realisation. At iteration k all nodes were synchronously triggered for update, followed by a sharing of these updated variables with local neighbours. The quadratic approximation matrix H_k^i was set to the scaled identity matrix $2I_i$ for each node i for all iterations.

We evaluated both synchronous QA-PDMM and PDMM using Matlab code on a Windows computer. In the implementation of synchronous



(a) Convergence rate of QS-PDMM and conventional PDMM for ridge regularized logistic regression (synchronous).

(b) Primal variable MSE versus local runtime of FS-PDMM and conventional PDMM for ridge regularized logistic regression (synchronous).

Figure 5.1: Performance of QA-PDMM and conventional PDMM.

PDMM, the L-BFGS algorithm [83] was used to solve local subproblems involving the functions $\{f_i(\boldsymbol{x}_i)|i \in \mathcal{V}\}$. For simplicity, we set all the matrices $P_{ij}\ \forall j \in \mathcal{V}_i,\ \forall i \in \mathcal{V}$ to be a constant scalar parameter ρ for both methods. At each iteration, the mean squared error (MSE) across all nodes in the graph

$$\text{MSE}^k = \frac{1}{|\mathcal{V}|}\sum_{i\in\mathcal{V}} \|\boldsymbol{x}_i^k - \boldsymbol{x}^\star\|^2,$$

was measured, where the global optimal solution $\boldsymbol{x}^\star$ was computed beforehand. Additionally, the cumulative computation time of all local subproblem optimizations were recorded in order to compare the computational efficiency of both algorithms. The optimal tuning parameter was found to be $\rho = 2.4$ for both methods, and this was used for the performance figures.

Fig. 5.1a shows the MSE as a function of iterations per node for RRLR using QA-PDMM and conventional PDMM. We see that QA-PDMM takes roughly twice as long as conventional PDMM to converge to the same ac-

curacy, with QA-PDMM requiring around ten iterations per node more than conventional PDMM to reach a MSE of 10^{-6}. In Fig. 5.1b we see that QA-PDMM has a significantly lower runtime than conventional PDMM at nearly two orders of magnitude faster, with conventional PDMM in this case using the L-BFGS algorithm [83] for local subproblem solution. From the experiments it appears that a relatively simple setting for the quadratic approximation matrix of $2I_i$ resulted in a stable and efficient QA-PDMM convergence path. Further experiments on the optimal setting for the approximation matrix H_i^k would be beneficial in determining the practical robustness of QA-PDMM in a variety of optimization contexts.

5.4 Summary

In this section we have presented an inexact synchronous update variation of PDMM named Quadratically Approximated PDMM (QA-PDMM). The algorithm performs a second order approximation of local cost functions to improve the computational efficiency of local update iterations. We provide two convergence analyses for QA-PDMM, with the only necessary condition for convergence being that the quadratic approximation parameter matrix at each node must be larger than the Lipschitz constant of each node's cost function. We demonstrate experimentally that QA-PDMM converges approximately half as slowly as conventional PDMM for the case of partial consensus ridge regularized logistic regression over a random graph, while reducing the local subproblem solution runtime by over an order of magnitude. As future work a closer study on setting the quadratic approximation matrix would be beneficial, both analytically and experimentally.

Chapter 6

FSQA-PDMM

So far we have introduced two separate algorithms based on PDMM that help facilitate efficient distributed processing for wireless sensor networks: Function Splitting PDMM (FS-PDMM) for problems with regularization functions; and Quadratically Approximated PDMM (QA-PDMM) for problems with Lipschitz smooth cost functions. Both demonstrate significant local computational savings when performing distributed optimization, with minor reductions in global network convergence times. In many cases they also eliminate the need for costly optimization package overheads. In this chapter we will combine the two approaches to develop FSQA-PDMM, which may be applied when dealing with local functions that are the sum of a smooth cost function term and a non-smooth regularization term.

We begin by presenting the FSQA-PDMM algorithm and summarizing the conditions required for convergence. We then present the convergence analysis for synchronous FSQA-PDMM, using analysis similar to FS-PDMM and QA-PDMM, and show that these lead to a guaranteed sublinear convergence rate of $O(1/k)$. Finally, we present simulated experiments that confirm the effectiveness of FSQA-PDMM when applied to a distributed sparse ridge regularized logistic regression problem over a general random graph with partial consensus.

6.1 The FSQA-PDMM Algorithm

To begin, we assume that over our network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ we wish to perform regularized optimization of the form

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{V} \big(f_i(\boldsymbol{x}_i) + R_i(\boldsymbol{M}_i \boldsymbol{x}_i)\big) \\ \text{subject to} \quad & \boldsymbol{A}_{i|j}\boldsymbol{x}_i = \boldsymbol{A}_{j|i}\boldsymbol{x}_j \quad \forall (i,j) \in \mathcal{E}, \end{aligned} \tag{6.1}$$

where $\boldsymbol{M}_i \in \mathbb{R}^{\boldsymbol{M}_i \times n_i}$ defines a general linear mapping on the variable $\boldsymbol{x}_i$ prior to regularization, the loss function $f_i : \mathbb{R}^{n_i} \to \mathbb{R} \cup \{+\infty\}$ and regularization function $R_i : \mathbb{R}^{\boldsymbol{M}_i} \to \mathbb{R} \cup \{+\infty\}$ at node i are both closed, proper, and convex, the matrices $\{\boldsymbol{A}_{i|j}, \boldsymbol{A}_{j|i}\}$ define a linear constraint across the edge (i, j). As in chapter 5, we make the additional assumption that each convex cost function $f_i : \mathbb{R}^{n_i} \to \mathbb{R}$ is continuously differentiable with the Lipschitz continuous gradient $Q_i > 0$:

$$\|\nabla f_i(\boldsymbol{x}_i) - \nabla f_i(\boldsymbol{y}_i)\| \leq Q_i \|\boldsymbol{x}_i - \boldsymbol{y}_i\| \quad \forall \boldsymbol{x}_i, \boldsymbol{y}_i \in \mathbb{R}^{n_i}, \tag{6.2}$$

As in chapter 4 we split the cost and regularization functions and embed them in a larger virtual network with $2V$ nodes. This allows the simplification of local primal variable updates, including the use of proximal operators when performing updates involving the regularization function R_i. We are then able to apply the same reasoning as in chapter 5 to approximate the cost functions f_i at each node i using local gradient information, leading to additional simplification of updates involving the cost function.

FSQA-PDMM, as summarized in Algorithm 3, requires the solution of a linear system for the primal variable $\boldsymbol{x}_i$ and a proximal update for the primal variable $\boldsymbol{z}_i$. The dual variables $\boldsymbol{\nu}_i$ and $\boldsymbol{\lambda}_{i|j}$ $\forall j \in \mathcal{V}_i$ are then updated using linear mappings. Note that, as with the QA-PDMM algorithm, the FSQA-PDMM algorithm is synchronous, requiring all nodes to update their variables simultaneously.

Algorithm 3 FSQA-PDMM

Initialize:

specify $f_i, R_i, \boldsymbol{M}_i, \boldsymbol{A}_{i|j}, \rho, \epsilon \qquad \forall i \in \mathcal{V}, \forall j \in \mathcal{V}_i$

randomly initialize $(\boldsymbol{x}_i, \boldsymbol{z}_i, \boldsymbol{\nu}_i, \boldsymbol{\lambda}_{i|j})\ \forall i \in \mathcal{V}, \forall j \in \mathcal{V}_i$

Update procedure:

while *stopping criterion* $==$ *false* **do**

trigger all nodes $i \in \mathcal{V}$ for synchronous update

$$\boldsymbol{x}_i^{k+1} = (\textstyle\sum_{j\in\mathcal{V}_i} \rho \boldsymbol{A}_{i|j}^T \boldsymbol{A}_{i|j} + \rho \boldsymbol{M}_i^T \boldsymbol{M}_i + H_i^k)^{-1} \Big[H_i^k \boldsymbol{x}_i^k - \nabla f_i(\boldsymbol{x}_i^k) + \textstyle\sum_{j\in\mathcal{V}_i} \boldsymbol{A}_{i|j}^T (\boldsymbol{A}_{j|i} \boldsymbol{x}_j^k - \boldsymbol{\lambda}_{j|i}^k) + \rho \boldsymbol{M}_i^T \boldsymbol{z}_i^k \Big]$$

$$\boldsymbol{z}_i^{k+1} = \mathbf{prox}_{R_i, 1/\rho}(2\boldsymbol{M}_i \boldsymbol{x}_i^k - \boldsymbol{z}_i^{k-1} + \boldsymbol{\nu}_i^{k-1}/\rho)$$

$$\boldsymbol{\nu}_i^{k+1} = \rho(2\boldsymbol{M}_i \boldsymbol{x}_i^k - \boldsymbol{z}_i^{k+1} - \boldsymbol{z}_i^{k-1}) + \boldsymbol{\nu}_i^{k-1}$$

$$\boldsymbol{\lambda}_{i|j}^{k+1} = \rho(\boldsymbol{A}_{j|i} \boldsymbol{x}_j^k - \boldsymbol{A}_{i|j} \boldsymbol{x}_i^{k+1}) - \boldsymbol{\lambda}_{j|i}^k \qquad \forall j \in \mathcal{V}_i$$

share $\boldsymbol{x}_i^{k+1}, \boldsymbol{\lambda}_{i|j}^{k+1}$ with neighbours $j \in \mathcal{V}_i$

if $\|\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k\| + 1/V_i \textstyle\sum_{j\in\mathcal{V}_i} \|\boldsymbol{\lambda}_{i|j}^{k+1} - \boldsymbol{\lambda}_{i|j}^k\| < \epsilon$ **then**

stopping criterion $=$ *true*

end if

$k \leftarrow k + 1$

end while

6.2 Convergence Analysis

In this section we will present a convergence analysis for the FSQA-PDMM algorithm. The analysis requires two main steps, building on the analyses of chapters 4 and 5. Firstly, we show that a single pass of the FSQA-PDMM algorithm may be seen as a conventional synchronous PDMM update across a network of size $2V$ where cost functions f_i and regularization functions R_i are separated over the extra network nodes. The cost functions f_i have additionally been quadratically approximated using local gradient information. We then perform a convergence analysis over this expanded network where a subset of the nodes perform an inexact update using the cost function approximation and the remaining nodes perform an exact update.

6.2.1 Equivalence to Mixed Approximation PDMM

We begin by providing a theorem, similar to Theorem 1:

Theorem 4. *FSQA-PDMM, as presented in Algorithm 3, is equivalent to performing a mixture of PDMM and QA-PDMM updates on a network of size $2V$.*

Proof. We begin by restating the FSQA-PDMM update iterations from Algorithm 3 using the definition of the proximal operator (4.4). At each iteration k we select a random node i and therefore perform

$$\begin{aligned}\boldsymbol{x}_i^{k+1} = \arg\min_{\boldsymbol{x}_i} \Bigg[& f_i(\boldsymbol{x}_i^k) + (\boldsymbol{x}_i - \boldsymbol{x}_i^k)^T \nabla f_i(\boldsymbol{x}_i^k) + \frac{1}{2}\|\boldsymbol{x}_i - \boldsymbol{x}_i^k\|_{H_i^k}^2 \\ & + \boldsymbol{x}_i^T \Big(\sum_{j \in \mathcal{V}_i} \boldsymbol{A}_{i|j}^T \boldsymbol{\lambda}_{j|i}^k + \boldsymbol{M}_i^T \boldsymbol{\nu}_i^k \Big) \\ & + \sum_{j \in \mathcal{V}_i} \frac{\rho}{2} \|\boldsymbol{A}_{i|j}\boldsymbol{x}_i - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^k\|_2^2 + \frac{\rho}{2}\|\boldsymbol{M}_i \boldsymbol{x}_i - \boldsymbol{z}_i^k\|_2^2 \Bigg] \end{aligned} \tag{6.3a}$$

$$\boldsymbol{\lambda}_{i|j}^{k+1} = \rho(\boldsymbol{A}_{j|i}\boldsymbol{x}_j^k - \boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1}) - \boldsymbol{\lambda}_{j|i}^k \quad \forall j \in \mathcal{V}_i \tag{6.3b}$$

$$\boldsymbol{\eta}_i^{k+1} = \rho(\boldsymbol{z}_i^k - \boldsymbol{M}_i \boldsymbol{x}_i^{k+1}) - \boldsymbol{\nu}_i^k \tag{6.3c}$$

$$\boldsymbol{z}_i^{k+1} = \arg\min_{\boldsymbol{z}_i} \left[\frac{\rho}{2} \|\boldsymbol{z}_i - (\boldsymbol{M}_i \boldsymbol{x}_i^k - \boldsymbol{\eta}_i^k/\rho)\|_2^2 + R_i(\boldsymbol{z}_i) \right] \tag{6.3d}$$

$$\boldsymbol{\nu}_i^{k+1} = \rho(\boldsymbol{M}_i \boldsymbol{x}_i^k - \boldsymbol{z}_i^{k+1}) - \boldsymbol{\eta}_i^k \tag{6.3e}$$

where we have introduced the extra intermediate variable $\boldsymbol{\eta}_i^{k+1}$ rather than including $\boldsymbol{z}_i^k$ explicitly in the minimization over $\boldsymbol{z}_i$, for notational convenience. Additionally, the minimization (6.3a) is equivalent to the $\boldsymbol{x}_i$ update in Algorithm 3. Using similar steps to those in Theorem 1, we see that these update iterations are equivalent to performing $\forall i \in \mathcal{V}$

$$\boldsymbol{x}_i^{k+1} = \arg\min_{\boldsymbol{x}_i} \left[f_i^k(\boldsymbol{x}_i) + \boldsymbol{x}_i^T \Big(\sum_{j \in \mathcal{U}_i} \boldsymbol{B}_{i|j}^T \boldsymbol{\lambda}_{j|i}^k \Big) + \sum_{j \in \mathcal{U}_i} \frac{\rho}{2} \|\boldsymbol{B}_{i|j} \boldsymbol{x}_i - \boldsymbol{B}_{j|i} \boldsymbol{x}_j^k\|_2^2 \right] \tag{6.4a}$$

$$\boldsymbol{\lambda}_{i|j}^{k+1} = \rho(\boldsymbol{B}_{j|i} \boldsymbol{x}_j^k - \boldsymbol{B}_{i|j} \boldsymbol{x}_i^{k+1}) - \boldsymbol{\lambda}_{j|i}^k \quad \forall j \in \{\mathcal{V}_i \cup t\} \tag{6.4b}$$

$$\boldsymbol{x}_t^{k+1} = \arg\min_{\boldsymbol{x}_t} \left[R_t(\boldsymbol{x}_t) + \boldsymbol{x}_t^T \boldsymbol{B}_{t|i}^T \boldsymbol{\lambda}_{i|t}^k + \frac{\rho}{2} \|\boldsymbol{B}_{t|i} \boldsymbol{x}_t - \boldsymbol{B}_{i|t} \boldsymbol{x}_i^k\|_2^2 \right] \quad t = i + V \tag{6.4c}$$

$$\boldsymbol{\lambda}_{t|i}^{k+1} = \rho(\boldsymbol{B}_{i|t} \boldsymbol{x}_i^k - \boldsymbol{B}_{t|i} \boldsymbol{x}_t^{k+1}) - \boldsymbol{\lambda}_{i|t}^k \quad t = i + V \tag{6.4d}$$

where $f_i^k(\boldsymbol{x}_i)$ is the quadratically approximated function (5.4), the new constraint matrices $(\boldsymbol{B}_{i|j}, \boldsymbol{B}_{j|i})$ are the same as our original matrices $(\boldsymbol{A}_{i|j}, \boldsymbol{A}_{j|i})$ with the addition of two new constraints for each extra "virtual" node, i.e.,

$$\boldsymbol{B}_{i|j} = \boldsymbol{A}_{i|j} \quad \forall (i,j) \in \mathcal{E}, \tag{6.5}$$

$$\boldsymbol{B}_{i|t} = \boldsymbol{M}_i \quad \text{for } t = i + V, \tag{6.6}$$

$$\boldsymbol{B}_{t|i} = \boldsymbol{I}_i \quad \text{for } t = i + V, \tag{6.7}$$

and we have changed the index of our regularization function R_t to reflect the change of variable index. ■

As in chapter 4, we have embedded our problem in a "virtual" network with $2V$ nodes, where the topology is the same as our physical network

with the addition of an extra "virtual" node attached to each physical node i. However, our cost functions f_i are now quadratically approximated by f_i^k at each iteration and the updates are now performed synchronously rather than asynchronously.

6.2.2 Convergence of Mixed Approximation PDMM

To guarantee the convergence of the equivalent update iterations (6.4a)-(6.4d) for FSQA-PDMM, in this section we will prove a more general convergence result that applies to a network where an arbitrary number of nodes hold Lipschitz continuous cost functions and their associated primal updates have been approximated using (5.5)-(5.4). We refer to this more general problem as Mixed Approximation PDMM (MA-PDMM).

Disjoint subsets and optimality conditions

We begin by dividing our nodes into two disjoint subsets: those nodes whose functions are approximated, denoted by set $\mathcal{U}$; and those nodes whose functions are not approximated, denoted by set $\mathcal{W}$. We therefore have that the union of these subsets results in our original global node set, i.e., $\mathcal{U} \cup \mathcal{W} = \mathcal{V}$. At nodes in the approximation set, $i \in \mathcal{U}$, each convex function f_i is assumed to be continuously differentiable with the Lipschitz continuous gradient $L_i(f_i) > 0$, i.e.,

$$\|\nabla f_i(\boldsymbol{x}_i) - \nabla f_i(\boldsymbol{y}_i)\| \leq L_i(f_i)\|\boldsymbol{x}_i - \boldsymbol{y}_i\| \quad \forall \boldsymbol{x}_i, \boldsymbol{y}_i \in \mathbb{R}^{n_i}, \tag{6.8}$$

where $\|\cdot\|$ denotes the standard Euclidean norm. Each individual function f_i at iteration k may then be approximated as (see [100])

$$f_i^k(\boldsymbol{x}_i) = f_i(\boldsymbol{x}_i^k) + (\boldsymbol{x}_i - \boldsymbol{x}_i^k)^T \nabla f_i(\boldsymbol{x}_i^k) + \frac{L_i}{2}\|\boldsymbol{x}_i - \boldsymbol{x}_i^k\|^2 \;\; i \in \mathcal{U}, \tag{6.9}$$

where $L_i > 0$. Replacing $f_i(\boldsymbol{x}_i)$ in (2.35a)-(2.35b) with the approximation $f_i^k(\boldsymbol{x}_i)$ for the node set $\mathcal{U}$ gives the updating expressions

$$\boldsymbol{x}_i^{k+1} = \arg\min_{\boldsymbol{x}_i} \Big[f_i^k(\boldsymbol{x}_i) + \sum_{j \in \mathcal{N}_i} \frac{1}{2}\|\boldsymbol{A}_{i|j}\boldsymbol{x}_i - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^k\|^2_{\boldsymbol{P}_{ij}}$$

$$+\sum_{j\in\mathcal{N}_i}\boldsymbol{\lambda}_{j|i}^{k,T}\boldsymbol{A}_{i|j}\boldsymbol{x}_i\Big] \quad i\in\mathcal{U} \tag{6.10}$$

$$\boldsymbol{\lambda}_{i|j}^{k+1}=\boldsymbol{P}_{ij}(\boldsymbol{A}_{j|i}\boldsymbol{x}_j^k-\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1})-\boldsymbol{\lambda}_{j|i}^k \; i\in\mathcal{U}, j\in\mathcal{N}_i, \tag{6.11}$$

while nodes in the set without approximation, $\mathcal{W}$, retain the original exact PDMM update

$$\boldsymbol{x}_i^{k+1}=\arg\min_{\boldsymbol{x}_i}\Big[f_i(\boldsymbol{x}_i)+\sum_{j\in\mathcal{N}_i}\frac{1}{2}\|\boldsymbol{A}_{i|j}\boldsymbol{x}_i-\boldsymbol{A}_{j|i}\boldsymbol{x}_j^k\|^2_{\boldsymbol{P}_{ij}}$$
$$+\sum_{j\in\mathcal{N}_i}\boldsymbol{\lambda}_{j|i}^{k,T}\boldsymbol{A}_{i|j}\boldsymbol{x}_i\Big] \quad i\in\mathcal{W} \tag{6.12}$$

$$\boldsymbol{\lambda}_{i|j}^{k+1}=\boldsymbol{P}_{ij}(\boldsymbol{A}_{j|i}\boldsymbol{x}_j^k-\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1})-\boldsymbol{\lambda}_{j|i}^k \; i\in\mathcal{W}, j\in\mathcal{N}_i. \tag{6.13}$$

Synchronous PDMM converges to an optimal point $\lim_{k\to\infty}(\boldsymbol{x}^k,\boldsymbol{\lambda}^k)=(\boldsymbol{x}^\star,\boldsymbol{\lambda}^\star)$ for the problem (2.32) if and only if $(\boldsymbol{x}^\star,\boldsymbol{\lambda}^\star)$ satisfies the following optimality conditions [157, 100]

$$\sum_{j\in\mathcal{N}_i}\boldsymbol{A}_{i|j}^T\boldsymbol{\lambda}_{i|j}^\star\in\partial f_i(\boldsymbol{x}_i^\star) \qquad i\in\mathcal{V} \tag{6.14}$$

$$\boldsymbol{A}_{i|j}\boldsymbol{x}_i^\star=\boldsymbol{A}_{j|i}\boldsymbol{x}_j^\star \qquad (i,j)\in\mathcal{E} \tag{6.15}$$

$$\boldsymbol{\lambda}_{i|j}^\star+\boldsymbol{\lambda}_{j|i}^\star=0 \qquad (i,j)\in\mathcal{E}. \tag{6.16}$$

Additionally, with (6.9)-(6.11) the optimality condition for each $\boldsymbol{x}_i^{k+1}$ at nodes in the approximation set $\mathcal{U}$ can be derived as

$$\nabla f_i(\boldsymbol{x}_i^k)+L_i(\boldsymbol{x}_i^{k+1}-\boldsymbol{x}_i^k)=\sum_{j\in\mathcal{N}_i}\boldsymbol{A}_{i|j}^T\boldsymbol{\lambda}_{i|j}^{k+1} \quad i\in\mathcal{U}. \tag{6.17}$$

We will show in next section that if the set of parameters $\{L_i|i\in\mathcal{V}\}$ are properly chosen in (6.9), synchronous MA-PDMM converges to an optimal point.

Constructing an inequality for approximate node updates

We now present an inequality formed over the summation of all nodes in the approximated update set $\mathcal{U}$. We begin by deriving an inequality at

each node $i \in \mathcal{U}$ for each approximated primal variable update $\boldsymbol{x}_i^{k+1}$ in (6.10):

Lemma 5. *Let $L_i \geq L_i(f_i)$ in the approximation function (6.9). Then for any $\boldsymbol{x}_i \in \mathbb{R}^{n_i}$,*

$$f_i(\boldsymbol{x}_i) - f_i(\boldsymbol{x}_i^{k+1}) \geq \frac{L_i}{2}\|\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k\|^2 - L_i(\boldsymbol{x}_i - \boldsymbol{x}_i^k)^T(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k) + (\boldsymbol{x}_i - \boldsymbol{x}_i^{k+1})^T \sum_{j \in \mathcal{N}_i} \boldsymbol{A}_{i|j}^T \boldsymbol{\lambda}_{i|j}^{k+1}. \tag{6.18}$$

Proof. Let each f_i for $i \in \mathcal{U}$ be a continuously differentiable function with the Lipschitz continuous gradient $L_i(f_i)$. For any $L_i \geq L_i(f_i)$, a standard inequality, as presented in Lemma 2.3 of [6], is then

$$f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{y}) + (\boldsymbol{x} - \boldsymbol{y})^T \nabla f_i(\boldsymbol{y}) + \frac{L_i}{2}\|\boldsymbol{x} - \boldsymbol{y}\|^2 \ \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n.$$

From the above inequality and (6.9), it then follows that

$$\begin{aligned}
& f_i(\boldsymbol{x}_i) - f_i(\boldsymbol{x}_i^{k+1}) \\
\geq & f_i(\boldsymbol{x}_i) - f_i^k(\boldsymbol{x}_i^{k+1}) \\
\overset{(a)}{\geq} & f_i(\boldsymbol{x}_i^k) + (\boldsymbol{x}_i - \boldsymbol{x}_i^k)^T \nabla f_i(\boldsymbol{x}_i^k) - f_i^k(\boldsymbol{x}_i^{k+1}) \\
= & f_i(\boldsymbol{x}_i^k) + (\boldsymbol{x}_i - \boldsymbol{x}_i^k)^T \nabla f_i(\boldsymbol{x}_i^k) \\
& - \left[f_i(\boldsymbol{x}_i^k) + (\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k)^T \nabla f_i(\boldsymbol{x}_i^k) + \frac{L_i}{2}\|\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k\|^2 \right] \\
= & (\boldsymbol{x}_i - \boldsymbol{x}_i^{k+1})^T \nabla f_i(\boldsymbol{x}_i^k) - \frac{L_i}{2}\|\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k\|^2 \\
\overset{(b)}{=} & (\boldsymbol{x}_i - \boldsymbol{x}_i^{k+1})^T \left(\sum_{j \in \mathcal{N}_i} \boldsymbol{A}_{i|j}^T \boldsymbol{\lambda}_{i|j}^{k+1} - L_i(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k) \right) \\
& - \frac{L_i}{2}\|\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k\|^2,
\end{aligned}$$

where step (a) uses the property that f_i is a convex function and step (b) uses (6.17). By using algebra, the above expression can be further simplified as (6.18). ■

Constructing an inequality for exact node updates

We now present an inequality formed over the summation of all nodes in the exact update set $\mathcal{W}$. We begin by deriving an inequality at each node $i \in \mathcal{W}$ for each exact primal variable update $\boldsymbol{x}_i^{k+1}$ in (6.12):

Lemma 6. *For any $\boldsymbol{x}_i \in \mathbb{R}^{n_i}$ and $i \in \mathcal{W}$,*

$$f_i(\boldsymbol{x}_i) - f_i(\boldsymbol{x}_i^{k+1}) \geq (\boldsymbol{x}_i - \boldsymbol{x}_i^{k+1})^T \sum_{j \in \mathcal{N}_i} \boldsymbol{A}_{i|j}^T \boldsymbol{\lambda}_{i|j}^{k+1}. \tag{6.19}$$

Proof. We begin by using a standard inequality (as presented in the proof for Lemma 8 in the appendix of []) exploited in the convergence analysis for ADMM and PDMM. Let $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$ be two arbitrary closed, proper, and convex functions where $\boldsymbol{x}^*$ minimizes the sum of the two functions, i.e. $\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}}(f_1(\boldsymbol{x}) + f_2(\boldsymbol{x}))$. Then,

$$f_1(\boldsymbol{x}) - f_1(\boldsymbol{x}^*) \geq (\boldsymbol{x}^* - \boldsymbol{x})^T r(\boldsymbol{x}^*) \quad \forall \boldsymbol{x} \tag{6.20}$$

where $r(\boldsymbol{x}^*) \in \partial_{\boldsymbol{x}} f_2(\boldsymbol{x}^*)$. Applying (6.20) to the exact updating equation (6.12) yields

$$f_i(\boldsymbol{x}_i) - f_i(\boldsymbol{x}_i^{k+1}) \geq (\boldsymbol{x}_i - \boldsymbol{x}_i^{k+1})^T \sum_{j \in \mathcal{N}_i} \boldsymbol{A}_{i|j}^T \Big(\boldsymbol{P}_{ij} (\boldsymbol{c}_{ij} - \tag{6.21}$$

$$\boldsymbol{A}_{i|j} \boldsymbol{x}_i^{k+1} - \boldsymbol{A}_{j|i} \boldsymbol{x}_i^k) + \boldsymbol{\lambda}_{j|i}^k \Big), \tag{6.22}$$

and substituting in (6.12) gives (6.19) ∎

Combined network inequality and convergence properties

In this subsection, we derive the convergence properties of synchronous MA-PDMM based on Lemma 6. The derivation procedure is similar to our early work [157] for analyzing synchronous PDMM. With the inequalities presented in Lemmas 5 and 6, we may now derive an inequality over all nodes in our network to analyse the behaviour of mixed approximate and exact updates for PDMM.

Lemma 7. *Let $(\boldsymbol{x}^\star, \boldsymbol{\lambda}^\star)$ be an optimal solution satisfying (6.14)-(6.16). The updated estimate $(\boldsymbol{x}^{k+1}, \boldsymbol{\lambda}^{k+1})$ is obtained by performing (6.9)-(6.11) or (6.12)-(6.13), depending on whether the node belongs to set $\mathcal{U}$ or $\mathcal{W}$. Then there is*

$$\begin{aligned}
0 &\leq 2\sum_{i\in\mathcal{V}}\Big[f_i(\boldsymbol{x}_i^{k+1}) - f_i(\boldsymbol{x}_i^\star) - \boldsymbol{x}_i^{k+1,T}\sum_{j\in\mathcal{N}_i}\boldsymbol{A}_{i|j}^T\boldsymbol{\lambda}_{i|j}^{k+1}\Big] \\
&\leq \frac{1}{2}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\Big[\|\boldsymbol{P}_{ij}^{-1/2}(\boldsymbol{\lambda}_{i|j}^* + \boldsymbol{\lambda}_{j|i}^k) + \boldsymbol{P}_{ij}^{1/2}(\boldsymbol{A}_{ij}\boldsymbol{x}_i^* - \boldsymbol{A}_{ji}\boldsymbol{x}_j^k)\|^2 \\
&\quad - \|\boldsymbol{P}_{ij}^{-1/2}(\boldsymbol{\lambda}_{i|j}^* + \boldsymbol{\lambda}_{j|i}^{k+1}) + \boldsymbol{P}_{ij}^{1/2}(\boldsymbol{A}_{ij}\boldsymbol{x}_i^* - \boldsymbol{A}_{ji}\boldsymbol{x}_j^{k+1})\|^2 \\
&\quad - \|\boldsymbol{P}_{ij}^{-1/2}(\boldsymbol{\lambda}_{i|j}^{k+1} + \boldsymbol{\lambda}_{j|i}^k) + \boldsymbol{P}_{ij}^{1/2}(\boldsymbol{A}_{ij}\boldsymbol{x}_i^{k+1} - \boldsymbol{A}_{ji}\boldsymbol{x}_j^k)\|^2\Big] \\
&\quad + \sum_{i\in\mathcal{U}} L_i\|\boldsymbol{x}_i^k - \boldsymbol{x}_i^\star\|^2 - \sum_{i\in\mathcal{U}} L_i\|\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^\star\|^2
\end{aligned} \tag{6.23}$$

Proof. See appendix A.3. ■

Next we show that the estimates $(\boldsymbol{x}^{k+1}, \boldsymbol{\lambda}^{k+1})$ are always bounded using the results of Lemma 7.

Lemma 8. *Every pair of estimates $(\hat{\boldsymbol{x}}_i^{k+1}, \hat{\boldsymbol{\lambda}}_{i|j}^{k+1})$, $i \in \mathcal{V}$, $j \in \mathcal{V}_i$, $k \geq 0$, in Lemma 7 is upper bounded by a constant M under a squared error criterion:*

$$\Big\|\boldsymbol{P}_{ij}^{-\frac{1}{2}}(\boldsymbol{\lambda}_{i|j}^\star + \boldsymbol{\lambda}_{j|i}^{k+1}) + \boldsymbol{P}_{ij}^{\frac{1}{2}}(\boldsymbol{A}_{i|j}\boldsymbol{x}_i^\star - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^{k+1})\Big\|^2 \leq M. \tag{6.24}$$

Proof. One can first prove (6.24) for $k = 0$ by using (6.23). The inequality (6.24) for $k > 0$ can then be proved recursively. ■

Upon obtaining the results in Lemma 7 and 8, we are ready to present the convergence rate of synchronous MA-PDMM.

Theorem 5. *Let $(\boldsymbol{x}^k, \boldsymbol{\lambda}^k)$, $k = 1, \ldots, K$, be obtained by performing (6.9)-(6.11) under the condition that $L_i \geq L_i(f_i)$ for nodes $i \in \mathcal{U}$, and (6.12)-(6.13) for nodes $i \in \mathcal{W}$. The average estimate $(\bar{\boldsymbol{x}}^K, \bar{\boldsymbol{\lambda}}^K) = (\frac{1}{K}\sum_{k=1}^K \boldsymbol{x}^k, \frac{1}{K}\sum_{k=1}^K \boldsymbol{\lambda}^k)$ satisfies*

$$0 \leq \sum_{i\in\mathcal{V}}\Big[f_i(\bar{\boldsymbol{x}}_i^K) - f_i(\boldsymbol{x}_i^\star) - \bar{\boldsymbol{x}}_i^{K,T}\sum_{j\in\mathcal{V}_i}\boldsymbol{A}_{i|j}^T\boldsymbol{\lambda}_{i|j}^\star\Big] \leq \mathcal{O}\Big(\frac{1}{K}\Big) \tag{6.25}$$

$$\lim_{K\to\infty} \boldsymbol{A}_{i|j}\bar{\boldsymbol{x}}_i^K = \boldsymbol{A}_{j|i}\bar{\boldsymbol{x}}_j^K \quad \forall (i,j) \in \mathcal{E} \tag{6.26}$$

$$\lim_{K\to\infty} \bar{\boldsymbol{\lambda}}_{i|j}^K + \bar{\boldsymbol{\lambda}}_{j|i}^K = \boldsymbol{0} \quad \forall (i,j) \in \mathcal{E}. \tag{6.27}$$

Proof. The proof is similar to that for Theorem 2 in [157]. ■

The conditions $\{L_i \geq L_i(f_i) | i \in \mathcal{V}\}$ in Theorem 5 ensures that synchronous MA-PDMM possesses the same convergence rate as synchronous PDMM. There is no additional requirement on the matrix set $\mathcal{P}$ for M-PDMM to work.

Finally, since FSQA-PDMM is equivalent to performing (6.4a)-(6.4d), which is a specific case of (6.10)-(6.11), the convergence results of MA-PDMM apply to the FSQA-PDMM algorithm.

6.3 Numerical Simulation

In this section we present the results of simulations using FSQA-PDMM to perform sparse ridge regularized logistic regression (SRRLR) over a randomly connected graph under the partial consensus paradigm. The local data and related decision variable $\boldsymbol{x}_i$ are kept locally at node i for optimization, and each node i applies l_1 regularization to this local data. Neighbouring nodes $l \in \mathcal{V}_i$ may share some elements with node i while using their own local data for collaborative optimization. This requires consensus to be reached between *elements* of vector $\boldsymbol{x}_i$ and each neighbouring vector $\boldsymbol{x}_j$, *not* full vector consensus.

For SRRLR, the problem function at node i is

$$g_i^{\text{RRLR}}(\boldsymbol{x}_i) = \frac{1}{P_i}\sum_{p=1}^{P_i} \log[1 + \exp(-c_{ip}\boldsymbol{d}_{ip}^T\boldsymbol{x}_i)] + \|\boldsymbol{x}_i\|_2^2 + \|\boldsymbol{x}_i\|_1, \tag{6.28}$$

where each node holds P_i training points consisting of feature vector $\boldsymbol{d}_{ip}$ and binary label c_{ip} for $p = 1, \ldots, P_i$.

FSQA-PDMM performing SRRLR were both compared to conventional PDMM to determine the convergence rates and the runtimes. A randomly connected graph of 10 nodes was formed, each with between five and ten vector elements. Each node randomly shared a subset of these elements with their neighbours to initialise the consensus pattern for the current simulation realisation. At iteration k all nodes were triggered for update, followed by a sharing of these updated variables with local neighbours. The quadratic approximation matrix H_k^i was set to the scaled identity matrix $2I_i$ for each node i for all iterations.

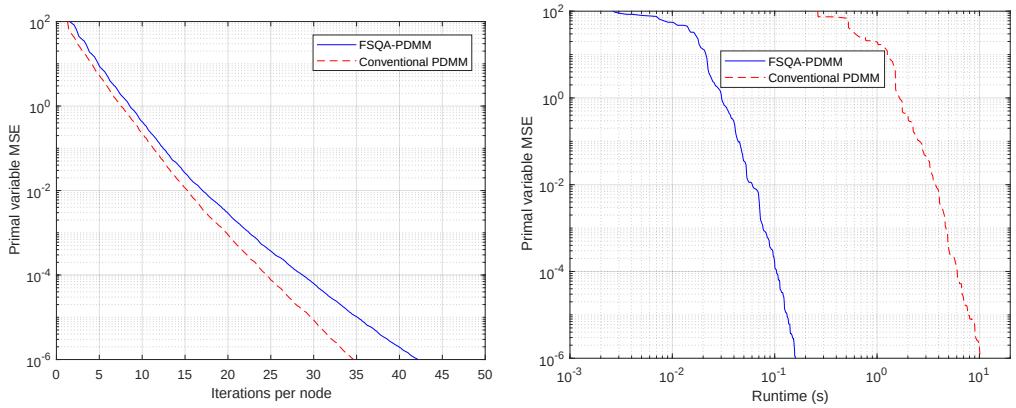
The optimal primal variable $\boldsymbol{x}_i^*$ was calculated for all nodes i to a precision of 10^{-9} prior to distributed optimization. At each iteration of the regularized and conventional PDMM optimization procedure we compute the mean-square error (MSE) across all nodes as

$$\text{MSE}^k = \frac{1}{V}\sum_{i\in\mathcal{V}}(\boldsymbol{x}_i^k - \boldsymbol{x}_i^{\text{opt}})^2, \tag{6.29}$$

and capture the runtime for each triggered node's update routine to complete.

Fig. 6.1a shows the MSE as a function of iterations per node for SRRLR using FSQA-PDMM and conventional PDMM. We see that they both perform similarly, with FSQA-PDMM requiring around five iterations per node more than conventional PDMM to reach an accuracy of 10^{-6}. However, from Fig. 6.1b we see that FSQA-PDMM has a runtime nearly two orders of magnitude less than conventional PDMM, which is performing local variable splitting and minimization using ADMM as in equations (4.3a)-(4.3c). This is due to the savings of function splitting combined with the computational savings of quadratic approximation.

We would like to reiterate that the FSQA-PDMM algorithm, while providing lower computational complexity and runtimes, also does not require any additional numerical optimizers. Practically, this allows our algorithm to be implemented in systems where processing units have low memory and are not able to store libraries of solvers. Large wireless sen-



(a) Convergence rate of FSQA-PDMM and conventional PDMM for elastic net regularized least-squares (asynchronous).

(b) Primal variable MSE versus local runtime of FSQA-PDMM and conventional PDMM for elastic net regularized least-squares (asynchronous).

Figure 6.1: Performance of FSQA-PDMM and conventional PDMM.

sor networks, for example, often have memory and processor restrictions at each node that may make this overhead requirement infeasible.

6.4 Summary

Combining the two distributed optimization algorithms FS-PDMM and QA-PDMM that employ function splitting and quadratic approximation, respectively, we have created FSQA-PDMM with the aim of further reducing local subproblem computation times. We proved that FSQA-PDMM converges at the rate $O(1/k)$ for a common class of functions given a condition on the quadratic approximation matrix and simulated experiments to show that local subproblems for sparse logistic regression run roughly two orders of magnitude faster than conventional PDMM, without the need for additional numerical optimizers. As with FS-PDMM and QA-PDMM, a reduction in convergence time was observed but this was negligible when compared with the computational savings. As further work, we intend

to prove the convergence of random and asynchronously performed node updates rather than the current synchronous paradigm.

Chapter 7

Finite Time Convergence PDMM

In chapters 4, 5, and 6 we discussed algorithms that reduce local computational complexity and often eliminate the need for optimization package overheads when performing distributed optimization over networks. As a trade-off, these methods slightly increased the number of network iterations required for a given convergence accuracy when compared with conventional PDMM. In this chapter we consider reducing the total workload of PDMM from the opposite approach, increasing the network overhead to provide significant reductions to the required network iterations for convergence. More specifically, we present an algorithm that converges in a finite number of iterations for the problem of quadratic distributed consensus, which we will refer to as Finite-Time PDMM (FT-PDMM). The algorithm requires a parameter setting protocol to be run prior to optimization that must be performed over a directed acyclic graph (DAG) reduced from the undirected general graph of the physical network. The theoretical work of this section was carried out solely by Dr. G. Zhang, and therefore will be omitted from this thesis. The author's contribution is an evaluation of the convergence of FT-PDMM when applied to quadratic consensus optimization. We will therefore present the resulting algorithms of this work and the experimental simulations that were used to verify and test these.

We begin with a summary of the quadratic problem form, specific no-

tation that will be assumed for this chapter, and provide the conventional PDMM synchronous updating scheme to solve this problem. We will then present a forward-backward PDMM updating scheme as well as the parameter setting and network routing overheads required for operation. Finally, we will present the experimental simulations that confirm the effectiveness of the algorithm while also hinting at the possibility of a non-routed synchronous PDMM approach that converges in finite time.

7.1 Problem Form and Conventional PDMM

Throughout the thesis so far we have dealt exclusively with the undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ used to describe the network of nodes over which our optimization is performed. In this chapter we also consider a directed graph $\vec{\mathcal{G}} = (\mathcal{V}, \vec{\mathcal{E}})$, where $\vec{\mathcal{E}} = \{[i,j] | i,j \in \mathcal{V}\}$ represents the set of all directed edges. The directed edge $[i,j]$ indicates that node i can reach node j through their edge, but the reverse is not true. We define a path from node i to j as a sequence of consecutive directed edges connecting i and j. The distance for a path from i to j measures the number of the consecutive edges in between. $\vec{\mathcal{G}} = (\mathcal{V}, \vec{\mathcal{E}})$ is a directed acyclic graph (DAG) if there exists no path starting and ending at the same node in the graph. As will be explained in subsection 7.2, any undirected cyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ can be reduced to a DAG by assigning proper directions for the edges in $\mathcal{E}$.

Note that every node i in a DAG $\vec{\mathcal{G}} = (\mathcal{V}, \vec{\mathcal{E}})$ may have incoming and outgoing edges. Therefore, we use $\mathcal{V}_{i,in}$ and $\mathcal{V}_{i,out}$ to represent the sets of preceding and succeeding neighbours of node i, respectively. It is immediate that $\mathcal{V}_i = \mathcal{V}_{i,in} \cup \mathcal{V}_{i,out}$ where $\mathcal{V}_i$ is the neighbourhood set of i in the corresponding undirected graph. Since the graph $\vec{\mathcal{G}}$ is directed and acyclic, there exists one or more nodes that do not have a preceding neighbour. We refer to these nodes as the *leaf* nodes. Finally, suppose there exists a path from node u to v. Node u is an *ancestor* of node v while conversely node v is a *descendant* of node u. We will use the DAG $\vec{\mathcal{G}}$ for the parameter selection

of FT-PDMM in section 7.2.

As a special form of (2.32), the quadratic consensus optimization over an undirected cyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is defined as

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{V} \left(f_i(\boldsymbol{x}_i) = \frac{1}{2} \boldsymbol{x}_i^T \boldsymbol{\Sigma}_i \boldsymbol{x}_i - \boldsymbol{a}_i^T \boldsymbol{x}_i \right) \\ \text{subject to} \quad & \boldsymbol{x}_i = \boldsymbol{x}_j \quad \forall (i,j) \in \mathcal{E}, \end{aligned} \tag{7.1}$$

where $\boldsymbol{\Sigma}_i \in \mathbb{R}^{n \times n}\ \forall i \in \mathcal{V}$ is a symmetric positive definite matrix (i.e., $\boldsymbol{\Sigma}_i \succ 0$), $\boldsymbol{a}_i \in \mathbb{R}^n$ is a constant vector at each node i, and $\boldsymbol{x} = [\boldsymbol{x}_1^T, \boldsymbol{x}_2^T, \ldots, \boldsymbol{x}_{|\mathcal{V}|}^T]^T \in \mathbb{R}^{|\mathcal{V}|n}$. It is clear that there exists a unique optimal point $\boldsymbol{x}^\star$ of the above quadratic optimization (7.1), where the distributed vectors $\{\boldsymbol{x}_i^\star | i \in \mathcal{V}\}$ take the form of

$$\boldsymbol{x}_i^\star = \left(\sum_{i \in \mathcal{V}} \boldsymbol{\Sigma}_i \right)^{-1} \left(\sum_{i \in \mathcal{V}} \boldsymbol{a}_i \right) \quad \forall i \in \mathcal{V}. \tag{7.2}$$

One method for obtaining the optimal point (7.2) is to apply an information aggregation approach [108, 28]. To do so, the graph $\mathcal{G}$ must first be converted to a tree-structured graph rooted at the fusion center. Information of the quadratic matrices $\{\boldsymbol{\Sigma}_i | i \in \mathcal{V}\}$ and linear vectors $\{\boldsymbol{a}_i | i \in \mathcal{V}\}$ may then be aggregated through the tree to the fusion center.

Applying conventional synchronous PDMM to problem (7.1) gives the explicit primal and dual update iterations for all $i \in \mathcal{V}$

$$\boldsymbol{x}_i^{k+1} = \left(\boldsymbol{\Sigma}_i + \sum_{j \in \mathcal{V}_i} \boldsymbol{P}_{ij} \right)^{-1} \left(\boldsymbol{a}_i + \sum_{j \in \mathcal{V}_i} \boldsymbol{\lambda}_{j|i}^k \right) \tag{7.3}$$

$$\boldsymbol{\lambda}_{i|j}^{k+1} = 2\boldsymbol{P}_{ij} \boldsymbol{x}_i^{k+1} - \boldsymbol{\lambda}_{j|i}^k \qquad \forall j \in \mathcal{V}_i, \tag{7.4}$$

As with the scalar tuning parameter of ADMM [15], the set $\mathcal{P}$ of positive definite matrices has a large impact on the convergence speed of PDMM. In [158] the effect of parameter tuning on the convergence speed of PDMM for the distributed averaging problem was experimentally investigated. A

more thorough theoretical study was performed in [161], where the optimal matrix set $\mathcal{P}^*$ was constructed for a quadratic consensus problem over a tree-structured graph, leading to finite-time convergence. In the following section we will present a method to construct the optimal matrix set $\mathcal{P}^*$ on our reduced DAG $\vec{\mathcal{G}}$ as well as the FT-PDMM algorithm that uses these parameters to perform finite-time optimization of the quadratic consensus problem (7.1).

7.2 Parameter Selection and FT-PDMM

Performing FT-PDMM requires the initial computation of the optimal parameter set $\mathcal{P}^*$ using a two step process. Firstly we convert an undirected cyclic graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ to a DAG $\vec{\mathcal{G}} = \{\mathcal{V}, \vec{\mathcal{E}}\}$. We then construct the matrices in $\mathcal{P}^*$ recursively over the directed edges in $\vec{\mathcal{E}}$ by using the quadratic matrices $\{\mathbf{\Sigma}_i | i \in \mathcal{V}\}$ of (7.1).

Suppose $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ is an undirected cyclic graph. We must assign a direction to every edge $(i, j) \in \mathcal{E}$ in order to build a DAG $\vec{\mathcal{G}} = (\mathcal{V}, \vec{\mathcal{E}})$. We first select any node $r \in \mathcal{V}$ to be a root node in the graph. Our objective is to construct a set $\vec{\mathcal{E}}$ such that starting from any node $i \in \mathcal{V}$ and traveling over the directed graph $\vec{\mathcal{G}}$ would eventually arrive at the root node r without revisiting any node on the way. With the root node r, we now consider constructing a proper set $\vec{\mathcal{E}}$ for the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. One approach to construct the set $\vec{\mathcal{E}}$ is to use the depth-first-search (DFS) method [118] to explore the graph $\mathcal{G}$ from the root node r. The edges within the graph may then be oriented toward the current search node as the DFS progresses from the root node outwards. Naturally, the root node r is a descendant of all the other nodes in the graph and all other nodes are ancestors of the root node r.

With the DAG we may now construct the matrices in $\mathcal{P}^*$ over $\vec{\mathcal{G}} = (\mathcal{V}, \vec{\mathcal{E}})$, starting from the leaf nodes. Suppose node u has no preceding neighbours, i.e., $\mathcal{V}_{u,in} = \{\}$. The matrix $\boldsymbol{P}_{uv}$ for an outgoing edge $[u, v]$ is

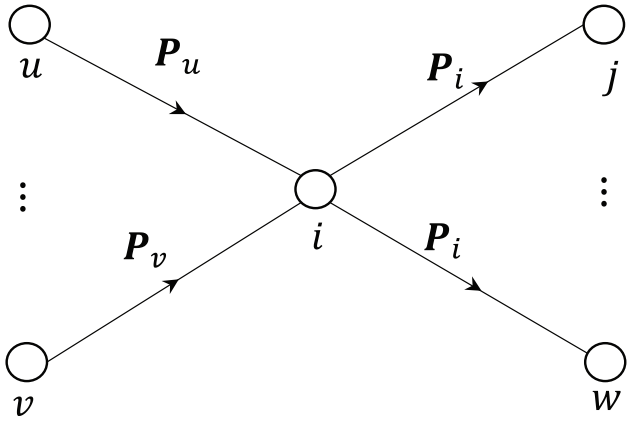


Figure 7.1: Recursive matrix construction from all leaf nodes towards the root node r over a DAG. The matrices over the outgoing edges of each node i are the same, i.e., $\boldsymbol{P}_{ij} = \boldsymbol{P}_i$ for all $j \in \mathcal{V}_{i,out}$.

built as

$$\boldsymbol{P}_{uv} = \boldsymbol{P}_u = \frac{1}{|\mathcal{V}_{u,out}|}\boldsymbol{\Sigma}_u \quad v \in \mathcal{V}_{u,out}, \tag{7.5}$$

where we introduce a new matrix $\boldsymbol{P}_u$ for node u. After computing (7.5), the other matrices in $\mathcal{P}^*$ may then be constructed recursively along the directed edges in $\vec{\mathcal{E}}$, towards the root node r. Suppose we are at node i to determine the matrices $\{\boldsymbol{P}_{ij}|j \in \mathcal{V}_{i,out}\}$. These outgoing edge matrices may then be computed using the matrices $\{\boldsymbol{P}_{ui}|u \in \mathcal{V}_{i,in}\}$ of the preceding edges as

$$\boldsymbol{P}_{ij} = \boldsymbol{P}_i = \frac{1}{|\mathcal{V}_{i,out}|}\left(\boldsymbol{\Sigma}_i + \sum_{u\in\mathcal{V}_{i,in}} \boldsymbol{P}_{ui}\right) \quad j \in \mathcal{V}_{i,out}, \tag{7.6}$$

where the matrix $\boldsymbol{P}_i$ is owned by node i (see Fig. 7.1 for demonstration). Using the fact that the quadratic matrices $\{\boldsymbol{\Sigma}_i, i \in \mathcal{V}\}$ in (7.1) are positive definite, it is immediate that the matrices obtained by (7.5)-(7.6) are also positive definite, which satisfy the conditions in (7.3)-(7.4). Equ. (7.5)-(7.6) together collect and propagate the information of all the quadratic matrices $\{\boldsymbol{\Sigma}_i|i \in \mathcal{V}\}$ along the directed edges towards the root node r.

With the optimal parameter set, we may now propose a forward-backward updating scheme to produce the optimal estimates $\{\boldsymbol{x}_i^\star | i \in \mathcal{V}\}$ of all nodes in finite time. Note that the optimal solution $\boldsymbol{x}_i^\star = \boldsymbol{x}_r^\star$ for all $i \in \mathcal{V}$. The FT-PDMM algorithm first computes $\boldsymbol{x}_r^\star$ using a sequential update ordering, starting from the leaf nodes and working towards the root node r (the *forward* portion of the update). Once the root node is updated, it obtains and then pushes the optimal point $\boldsymbol{x}_r^\star$ from the root node r along all the reverse direction of the edges in $\vec{\mathcal{E}}$ (the backward portion of the update).

Algorithm 4 summarizes the FT-PDMM update procedure. Forward computation activates nodes sequentially from $i = 1$ until $i = m$, and backward computation activates nodes in reverse order from $i = m - 1$ until $i = 1$. After the backward computation each node i holds a copy of the optimal point $\boldsymbol{x}_r^\star$. In the following section we will provide experimental verification for the FT-PDMM algorithm when applied to the simple case of distributed averaging. Additionally we explore the application of conventional synchronous PDMM when using the parameter set $\mathcal{P}^*$.

Algorithm 4 FT-PDMM

Initialize:

specify $\boldsymbol{\Sigma}_i$ and $a_i \quad \forall i \in \mathcal{V}$

construct the optimal parameter set $\mathcal{P}^*$ using $(7.5) - (7.6)$

randomly initialize $(\boldsymbol{x}_i, \boldsymbol{\lambda}_{i|j}) \qquad \forall i \in \mathcal{V}, \forall j \in \mathcal{V}_i$

Forward computation:

for node $i = 1, 2, \ldots, m-1$ **do**

at each iteration k, all nodes i synchronously perform

$$\boldsymbol{x}_i^1 = (2|\mathcal{V}_{i,out}|\boldsymbol{P}_i)^{-1}\left(\boldsymbol{a}_i + \sum_{u\in\mathcal{V}_{i,in}} \boldsymbol{\lambda}_{u|i}^1 + \sum_{v\in\mathcal{V}_{i,out}} \boldsymbol{\lambda}_{v|i}^0\right)$$

$$\boldsymbol{\lambda}_{i|v}^1 = 2\boldsymbol{P}_i\boldsymbol{x}_i^1 - \boldsymbol{\lambda}_{v|i}^0, v \in \mathcal{V}_{i,out}$$

end for

for node $i = r = m$ **do**

$$\boldsymbol{x}_r^1 = \left(\boldsymbol{\Sigma}_r + \sum_{u\in\mathcal{V}_{r,in}} \boldsymbol{P}_u\right)^{-1}\left(\boldsymbol{a}_r + \sum_{u\in\mathcal{V}_{r,in}} \boldsymbol{\lambda}_{u|r}^1\right)$$

$$\boldsymbol{\lambda}_{r|u}^2 = 2\boldsymbol{P}_u\boldsymbol{x}_r^1 - \boldsymbol{\lambda}_{u|r}^1, u \in \mathcal{V}_{r,in}$$

end for

Backward computation:

for node $i = m-1, \ldots, 1$ **do**

at each iteration k, all nodes i synchronously perform

$$\boldsymbol{x}_i^2 = (2|\mathcal{V}_{i,out}|\boldsymbol{P}_i)^{-1}\left(\boldsymbol{a}_i + \sum_{u\in\mathcal{V}_{i,in}} \boldsymbol{\lambda}_{u|i}^1 + \sum_{v\in\mathcal{V}_{i,out}} \boldsymbol{\lambda}_{v|i}^2\right)$$

$$\boldsymbol{\lambda}_{i|u}^2 = 2\boldsymbol{P}_u\boldsymbol{x}_i^2 - \boldsymbol{\lambda}_{u|i}^1, u \in \mathcal{V}_{i,in}$$

end for

7.3 Numerical Simulation

In this section we evaluate both the FT-PDMM algorithm and conventional synchronous PDMM using the optimal parameter set $\mathcal{P}^*$ for a distributed averaging problem. While the FT-PDMM algorithm should theoretically converge in finite iterations, the affect of numerical quantization is unaccounted for. Additionally, using the optimal parameter set $\mathcal{P}^*$ for synchronous PDMM may prove to provide new insights into the study of finite-time convergence.

The quadratic consensus problem (7.1) includes the distributed averaging problem as a special case. By letting $\boldsymbol{\Sigma}_i = \boldsymbol{I}_{n\times n}\ \forall i \in \mathcal{V}$, it is immediate that the optimal subvector $\boldsymbol{x}_i^\star$, $\forall i \in \mathcal{V}$, is the average of the vectors $\{\boldsymbol{a}_i | \forall i \in \mathcal{V}\}$. A 5×5 planar grid with directed edges as in Figure 7.2a and 7.2b were simulated, with random data vectors of dimension $n = 5$ generated at each node. The averaging operation of these data vectors can be formulated by solving the optimization problem (7.1) where the quadratic matrices $\{\boldsymbol{\Sigma}_i = \mathbf{I} | i \in \mathcal{V}\}$, and the vectors $\{\boldsymbol{a}_i | i \in \mathcal{V}\}$ are our random data vectors we wish to average.

In the first experiment, the DAG in Figure 7.2a was simulated, where the longest path is of distance 12. The corresponding matrix set $\mathcal{P}$ was obtained by following (7.5)-(7.6). Figure 7.3 displays the convergence results of each node's primal variable for a single instance of distributed averaging. After 25 iterations, the root node is the first to obtain the exact solution to the averaging problem. For every iteration thereafter, the backward computation of the updating scheme allows ancestor nodes back through the network to compute the optimal value as well. After 49 iterations, the leaf node is the last to obtain the optimal solution. The effect of numerical error on the algorithmic convergence can be ignored for a graph with a bounded number of nodes.

In the second experiment, we investigated the convergence speed of synchronous PDMM. Three setups for the matrix set $\mathcal{P}$ were tested for

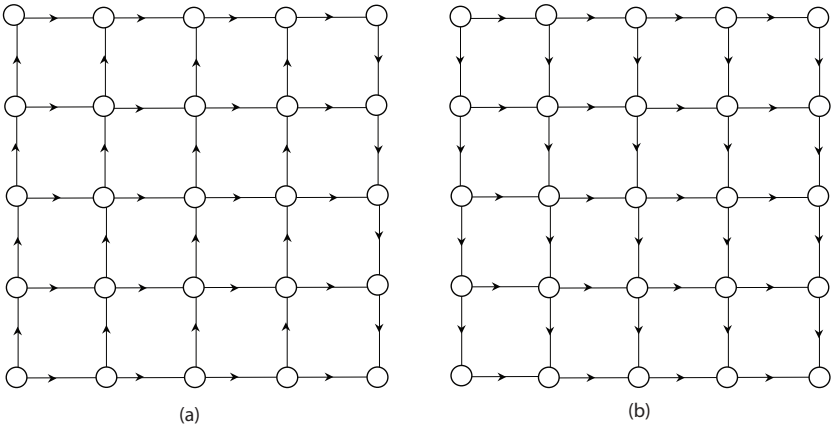


Figure 7.2: Two different DAGs for a 5×5 planar grid. The two DAGs share the same root node r located at the bottom-right corner.

synchronous PDMM: 1) the set $\mathcal{P}$ by following (7.5)-(7.6) for the DAG in Fig. 7.2a; 2) the set $\mathcal{P}$ by following (7.5)-(7.6) for the DAG in Fig. 7.2b; 3) the set $\mathcal{P} = \{\boldsymbol{P}_{ij} = \rho = 0.78 | (i,j) \in \mathcal{E}\}$. In the 3rd setup, all the matrices in $\mathcal{P}$ were set to be a constant value. The finite convergence of FT-PDMM is also included for reference. The parameter $\rho = 0.78$ leads to the fastest convergence speed of synchronous PDMM over all other constant values with respect to the mean squared error (MSE) criterion $\frac{1}{|\mathcal{V}|}\|\boldsymbol{x}^k - \boldsymbol{x}^*\|^2$. This was found by performing a parameter search for ρ over the range $(0, 10]$ in increments of 0.1 initially, and then increments of 0.01 over the range $[0.7, 0.8]$.

Figure 7.4a presents the MSE convergence of the above three setups over all the nodes. Each curve is obtained by combining the results of 500 randomly realised data vector instances across the grid network. It is clear that the third setup, with the optimal constant ρ value, leads to the fastest convergence speed when compared with the first two setups where the matrices in $\mathcal{P}$ are not constant.

We also evaluated the performance of the three setups only at the root node r, as displayed in Fig. 7.4b. Interestingly, it is found that the second setup leads to finite-time convergence of the estimate at the root node r in

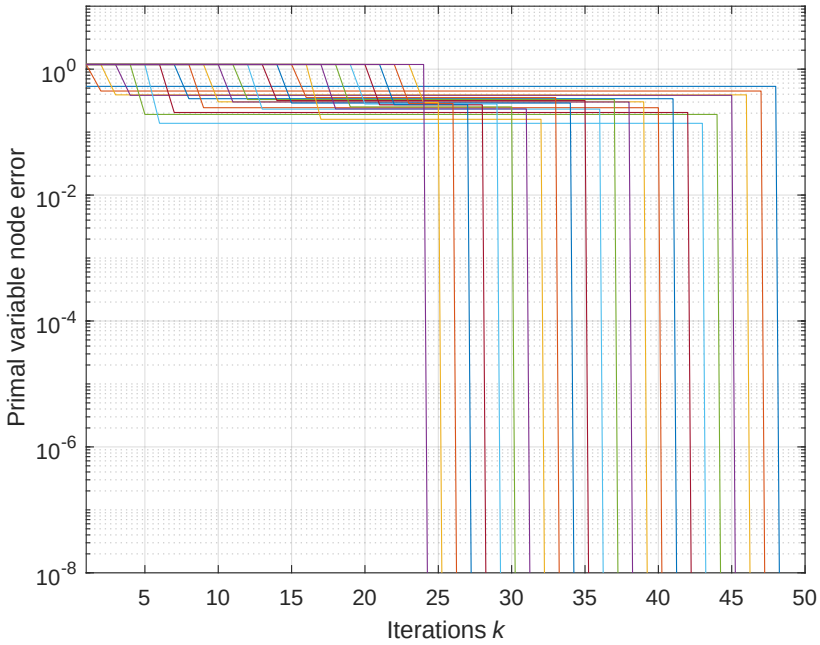
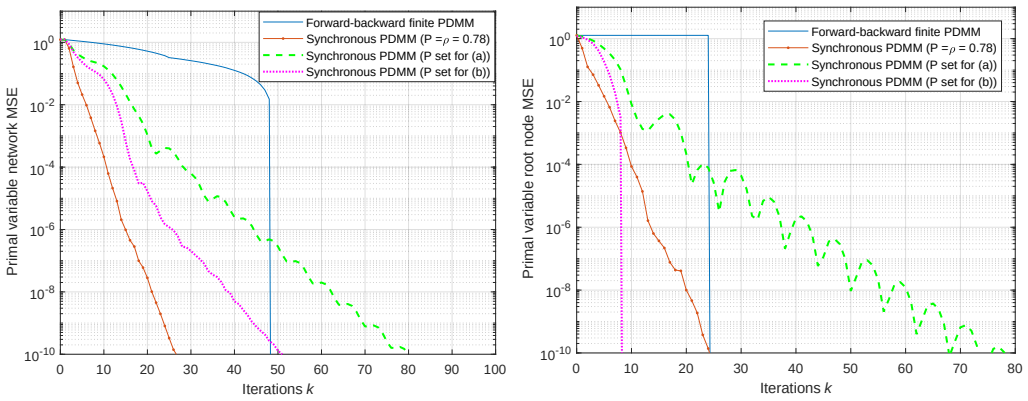


Figure 7.3: The squared error for each node versus iteration number k for the forward-backward updating scheme. The 25 curves correspond to the 25 nodes in the 5×5 planar grid.

9 iterations. This may be due to the property that the DAG in Fig. 7.2b has a symmetric topology. The 9 iterations coincide with the longest distance (equal to 8) between the leaf node and the root node r plus one in Fig. 7.2b. On the other hand, the first setup takes much more iterations to converge than the second and third setups. This may be due to the property that the DAG in Fig. 7.2a is not symmetric, which is likely to be more common in practice. As such, it appears the parameter-selection procedure (7.5)-(7.6) does not help with the convergence speed of synchronous PDMM for a general cyclic graph. Finally, the oscillation present for synchronous PDMM over DAG (b) shown in Fig. 7.2a appears to be the result of graph topology, although the exact cause of this extreme oscillation is unknown.



(a) Convergence rate of PDMM averaged over all network nodes.

(b) Convergence rate of PDMM at only the root node.

Figure 7.4: Performance of synchronous PDMM for three setups of the matrix set $\mathcal{P}$.

7.4 Summary

In this chapter we have considered the FT-PDMM algorithm using optimal parameter setting for solving the quadratic consensus problem over an undirected cyclic graph in finite iterations. The optimal parameters are selected by converting the undirected graph to a directed acyclic graph, which enables effective information flow over the directed graph, but adds an additional overhead to the processing procedure. We have shown that using these optimal parameters, the FT-PDMM algorithm converges to the optimal point in finite iterations. The FT-PDMM algorithm as well as conventional synchronous PDMM were experimentally simulated using the optimal parameter setting. The FT-PDMM algorithm was found to operate as expected, and interestingly the synchronous PDMM application was found to converge in finite iterations in a specific symmetric network topology. Future research directions include the extension of FT-PDMM to general decomposable convex optimizations where the objective functions are twice differentiable. It would be of interest to understand if the

Hessian matrices of the objective functions can be used to accelerate the convergence speed of PDMM.

Chapter 8

Signal Processing Applications

So far in this thesis we have developed four variations on the conventional PDMM algorithm with the specific goal of improving the flexibility and efficiency of distributed optimization in sensor networks. The simulated experiments verifying these algorithms have been relatively simple and general. In this section we present two potential real world applications in the area of distributed signal processing that are more complex and nuanced than the toy examples presented thus far. We first present a distributed beamforming algorithm that includes a sparsity regularization, allowing the algorithm to naturally select an optimal subset of nodes for combination. Secondly, we develop a distributed image fusion algorithm that may be used for imaging arrays, allowing overlapping imaging areas to be fused without a central processor.

8.1 Application to Distributed Beamforming

Until now, distributed acoustic beamforming has focused on optimizing for a beamformer over an entire network, with each node contributing to the beamformer output. We present a novel approach that introduces sparsity to this beamformer computation, where we attempt to optimize for a subset of nodes within the network that produce SNR gains roughly

equivalent to that of the optimal MVDR case. Due to the physical nature of sound, this approach trades a small loss in SNR for a large reduction in communication power and iterations required to produce a beamformer output by reducing the active node set of our network. Our approach operates in a fully distributed and asynchronous manner and does not require a high update iteration rate to produce an output at each sample.

The contributions of this section are a novel approach to distributed beamforming using sparsity, and a fully asynchronous and independent method for accomplishing this. Instead of computing a full network-wide MVDR beamformer we aim to find a subset of nodes and the associated beamformer that are optimal for a given sparsity level. We therefore describe a distributed sparse beamformer that approximates a full MVDR beamformer with a sparsity tradeoff parameter, using the asynchronous FS-PDMM algorithm from chapter 4 for distributed optimization. The traditional MVDR beamforming cost function is regularized with an l_1 penalty of the weight vector to encourage sparsity in the final optimized weight vector. This reduces the number of nodes that contribute to the final beamformer output, simplifying the aggregation step required at each time sample and greatly improving the practicality of the distributed beamforming system.

8.1.1 System model and background

We consider a network of nodes denoted by the set $\mathcal{V}$ with cardinality $V = |\mathcal{V}|$. The network is connected by a set of edges $\mathcal{E}$ with cardinality $E = |\mathcal{E}|$. If there exists an edge between two nodes i and j we say $(i, j) \in \mathcal{E}$. All nodes are considered self-connected. The node and edge sets together form our network graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The set of all nodes sharing an edge with a node i is denoted as its neighbourhood $\mathcal{V}_i$ with cardinality $V_i = |\mathcal{V}_i|$, i.e. $\mathcal{V}_i = \{l|(i, j) \in \mathcal{E}\}$.

We denote our wideband signal of interest as $s_0(t, \omega)$, with t and ω in-

dexing the time sample and frequency subband, respectively. Let $u_i(t,\omega)$ denote node i's observed noisy signal at time t and subband ω that is a combination of the desired signal $s_0(t,\omega)$, interferences $s_p(t,\omega)$ and additive noise $v_i(t,\omega)$. $d_{i0}(\omega)$ and $d_{ip}(\omega)$ represent the complex acoustic transfer functions that scale and phase shift each subband of the source and interference signals, respectively, which are constant over time. We may then express the linear receive model at node i of our source signal, P interferers, and noise as

$$\begin{aligned} u_i(t,\omega) &= d_{i0}(\omega)s_0(t,\omega) + \sum_{p=1}^{P} d_{ip}(\omega)s_p(t,\omega) + v_i(t,\omega), \\ &= d_i(\omega)s(t,\omega) + n_i(t,\omega). \end{aligned} \tag{8.1}$$

The noise and interference can be combined into the variable $n_i(t,\omega)$ for convenience, allowing us to drop the 0 subscripts on $d_{k0}(\omega)s_0(t,\omega)$. When multiple sensors, such as microphones, are used we may represent the acoustic transfer functions to each sensor as the vector $\boldsymbol{d}(\omega) \in \mathbb{C}^N$, allowing us to express a vector of noisy observations across these sensors as

$$\boldsymbol{u}(t,\omega) = \boldsymbol{d}(\omega)s(t,\omega) + \boldsymbol{n}(t,\omega), \tag{8.2}$$

where $\{\boldsymbol{u}(t,\omega), \boldsymbol{n}(t,\omega)\} \in \mathbb{C}^N$ are the vectors of observations and interferences from all N sensors, respectively.

Beamforming aims to estimate the source signal $s(t,\omega)$ by combining the noisy observations in a weighted sum to produce a single scalar output

$$z(t,\omega) = \boldsymbol{w}^H(\omega)\boldsymbol{u}(t,\omega), \tag{8.3}$$

where $z(t,\omega)$ is the beamformed output, $(\cdot)^H$ represents Hermitian transposition, and $\boldsymbol{w}(\omega) \in \mathbb{C}^N$ is the weighting vector used to combine the observations. Since the signal is assumed to be independent over each frequency bin ω and for all time t we will simplify notation by omitting these indices henceforth.

The traditional MVDR beamformer [133, 55] is obtained by minimizing the energy of the beamformed output signal subject to unity gain in the direction of the source, which may be expressed by the constrained optimization problem

$$\begin{aligned} &\text{minimize} \quad \mathrm{E}[|\boldsymbol{w}^H \boldsymbol{u}|^2], \\ &\text{subject to} \quad \boldsymbol{d}^H \boldsymbol{w} = 1, \end{aligned} \tag{8.4}$$

which can be solved using Lagrange multipliers as

$$\boldsymbol{w} = \frac{\boldsymbol{R}^{-1}\boldsymbol{d}}{\boldsymbol{d}^H \boldsymbol{R}^{-1}\boldsymbol{d}}, \tag{8.5}$$

where $\boldsymbol{R} = \mathrm{E}[\boldsymbol{n}\boldsymbol{n}^H]$ is the noise covariance matrix, and the source signal is assumed independent of the noise. The MVDR beamformer requires only knowledge of the source location (or equivalently the acoustic transfer functions $\boldsymbol{d}$ at each node) and knowledge of the noise statistics in the form of the covariance matrix $\boldsymbol{R}$.

Thoughout the remainder of the section we will assume a common mapping from the complex domain $\mathbb{C}^N$ to the real domain of twice the dimensionality $\mathbb{R}^{2N}$ in order to focus on the distributed convex optimization techniques employed. For the sake of brevity we omit the conversion of our problem, and for notational simplicity we will refer to our problem as one in $\mathbb{R}^N$. Additionally, we note that use of the l_1 norm to encourage sparsity, as presented in the next section, should therefore instead be the group sparse l_2 norm. This is because encouraging sparsity in the pairs of real values associated with our original complex signal is equivalent to the l_1 norm of our complex vectors. The derivations and algorithm remain the same, apart from the change of sparsity norm. For details see, for example, [4].

8.1.2 Distributed Sparse PDMM Beamformer

Previously, emphasis has been placed on creating fully distributed solutions to the traditionally centralized problem of MVDR beamforming.

These methods ideally aim to produce a beamformer output that is equal to the centralized case, but approximations are often introduced to deal with the restriction of the network topology or the nature of the distributed algorithm being used. The most common approximations are limitations on the structure of the noise covariance matrix, such as the requirement of diagonal dominance or a sparsity structure representative of the underlying network graph. However, as of the time of this work no approaches deal with optimizing for a sparse *weight vector*.

In very large networks, nodes distant from a source may practically capture none of the signal of interest, yet nearly all of the distributed beamforming methods reviewed require a mixing (or averaging) process across the entire network to produce a beamformer output at *each signal sample*. For networks on the order of 50 nodes (roughly the number used for simulation in the literature) the number of iterations required to reach convergence per sample is not a problem, but for network sizes where these distributed beamforming algorithms would actually be beneficial (at least on the order of thousands of nodes) the iteration time and associated communication cost required become infeasible.

One way of reducing the costly global mixing process is to encourage sparsity in our beamforming weight vector. Since those nodes closest to our source location are likely to be in the presence of a relatively high signal-to-noise ratio (due to the physical fall-off of sound power over distance), a sparsely optimized weight vector will naturally produce zero entries in nodes far from our source. Once we arrive at our sparse weight vector we may produce our beamformer output in one of three ways: we may perform the costly mixing process across the whole network; we may form a subnetwork with the nodes containing nonzero weight vector entries and perform a more efficient mixing over this far smaller network; or we may simply designate a single node (such as the node closest to the source) as our collection node for the current sample window and instruct all nodes with nonzero weight values to transmit to this collection node.

Since all nonzero nodes will, in most cases, be physically near to the source the transmission distances required will generally not be prohibitive.

Derivation of the sparse distributed PDMM beamformer

The proposed distributed sparse beamformer requires two separate operations. Firstly, the network will optimize (in a fully distributed manner) the weight values at all nodes while encouraging sparsity in these weights. This process will be independent and asynchronous at each node and requires communication only between neighbouring nodes. Additionally, this optimization process can be performed at any update rate without requiring network-wide convergence in a short time (such as between signal samples). Given these sparse weight values, we next require an aggregation operation to collect the weighted observations into a single scalar output. As mentioned previously, this may be done in multiple ways but we will restrict ourselves to single-node collection for the purpose of demonstrating the effectiveness of the sparse PDMM beamformer.

We begin by optimizing for a scaled version of the weight vector $\boldsymbol{x}^* = \boldsymbol{R}^{-1}\boldsymbol{d}$, where $(\cdot)^*$ refers to an optimal point, since the beamformer output is then simply the ratio of two averages [59]

$$z^* = \frac{\boldsymbol{x}^{*T}\boldsymbol{u}}{\boldsymbol{x}^{*T}\boldsymbol{d}} = \frac{\frac{1}{N}\sum_{k\in\mathcal{V}}[\boldsymbol{x}^*]_i^T[\boldsymbol{u}]_i}{\frac{1}{N}\sum_{k\in\mathcal{V}}[\boldsymbol{x}^*]_i^T[\boldsymbol{d}]_i}, \tag{8.6}$$

where $(\cdot)^T$ represents transposition and $[\cdot]_i$ denotes the kth element of a vector. Additionally, we would like to encourage sparsity in our optimized vector through the addition of an l_1 penalty on the resulting vector $\boldsymbol{x}$. Therefore, in order to obtain the optimal sparse vector $\boldsymbol{x}^*$ we construct the unconstrained, l_1-regularized quadratic program

$$\text{minimize} \quad f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{R}\boldsymbol{x} - \boldsymbol{d}^T\boldsymbol{x} + \alpha\|\boldsymbol{x}\|_1, \tag{8.7}$$

where α is our regularization parameter and $\|\cdot\|_1$ is the l_1 norm, widely known to encourage sparsity in our optimization solution [5, 117]. Next,

we make an approximating assumption that nodes more than two communication hops apart are uncorrelated allowing us to decompose the quadratic term (as in [97]), resulting in

$$\begin{aligned} \text{minimize} \quad & \sum_{k\in\mathcal{V}} \left(\frac{1}{2}\boldsymbol{x}_i^T \boldsymbol{R}_i \boldsymbol{x}_i - \boldsymbol{d}_i^T \boldsymbol{x}_i \right) \\ \text{subject to} \quad & \boldsymbol{A}_{i|j}\boldsymbol{x}_i + \boldsymbol{A}_{j|i}\boldsymbol{x}_l = 0 \quad \forall (i,j) \in \mathcal{E}, \end{aligned} \tag{8.8}$$

where the local vector $\boldsymbol{x}_i \in \mathbb{R}^{V_i}$ is node i's estimate of the elements of $\boldsymbol{x}$ belonging to its neighbourhood $\mathcal{V}_i$, $\boldsymbol{R}_i = (C^{\dagger 2} \circ \boldsymbol{R}) \in \mathbb{R}^{V_i \times V_i}$ is the covariance matrix for the neighbourhood $\mathcal{V}_i$ that we assume to be estimated within each neighbourhood, $\boldsymbol{d}_i \in \mathbb{R}^{V_i}$ is a vector containing all zeros apart from the ith entry that is equal to node i's scalar ATF d_i, the matrices $\boldsymbol{A}_{i|j} \in \mathbb{R}^{2\times V_i}$ and $\boldsymbol{A}_{j|i} \in \mathbb{R}^{2\times V_j}$ contain entries of 1, -1 or 0 to enforce consistency (with one row each for the consensus of node i and j's primary elements and their copies) and $C^{\dagger 2}$ is the protected elementwise inverse of the square of the adjacency matrix [54, 97].

Since the l_1 norm is separable across each element of $\boldsymbol{x}$ we may simply select out the scalar value $[\boldsymbol{x}]_i$ and apply a local l_1 regularization to it. Formulating this in a form consistent with FS-PDMM gives

$$\begin{aligned} \text{minimize} \quad & \sum_{k\in\mathcal{V}} \left(\frac{1}{2}\boldsymbol{x}_i^T \boldsymbol{R}_i \boldsymbol{x}_i - \boldsymbol{d}_i^T \boldsymbol{x}_i + \alpha_i \|\boldsymbol{M}_i \boldsymbol{x}_i\|_1 \right) \\ \text{subject to} \quad & \boldsymbol{A}_{i|j}\boldsymbol{x}_i = \boldsymbol{A}_{j|i}\boldsymbol{x}_l \quad \forall (i,j) \in \mathcal{E}, \end{aligned} \tag{8.9}$$

where the matrix $\boldsymbol{M}_i \in \mathbb{R}^{1\times V_i}$ selects out the value owned by node i, $[\boldsymbol{x}]_i$, and the local regularization parameter $\alpha_i = \alpha \ \forall i \in \mathcal{V}$. This leads to the asynchronous FS-PDMM update iterations $\forall i \in \mathcal{V}$

$$\begin{aligned} \boldsymbol{x}_i^{k+1} = & \Big(\sum_{j\in\mathcal{V}_i} \rho \boldsymbol{A}_{i|j}^T \boldsymbol{A}_{i|j} + \rho \boldsymbol{M}_i^T \boldsymbol{M}_i + \boldsymbol{R}_i \Big)^{-1} \Big(\boldsymbol{d}_i \\ & + \sum_{j\in\mathcal{V}_i} \boldsymbol{A}_{i|j}^T (\rho \boldsymbol{A}_{j|i} \boldsymbol{x}_j^k - \boldsymbol{\lambda}_{j|i}^k) + \boldsymbol{M}_i^T (\rho \boldsymbol{z}_i^k - \boldsymbol{\nu}_i^k) \Big) \\ \boldsymbol{z}_i^{k+1} = & \big(2\boldsymbol{M}_i \boldsymbol{x}_i^{k+1} - \boldsymbol{z}_i^k + \boldsymbol{\nu}_i^k/\rho - \alpha_i/\rho \big)_+ \end{aligned} \tag{8.10}$$

$$- \left(-2\boldsymbol{M}_i \boldsymbol{x}_i^{k+1} + \boldsymbol{z}_i^k - \boldsymbol{\nu}_i^k/\rho - \alpha_i/\rho\right)_+ \tag{8.11}$$

$$\boldsymbol{\nu}_i^{k+1} = \rho(2\boldsymbol{M}_i \boldsymbol{x}_i^{k+1} - \boldsymbol{z}_i^{k+1} - \boldsymbol{z}_i^k) + \boldsymbol{\nu}_i^k \tag{8.12}$$

$$\boldsymbol{\lambda}_{i|j}^{k+1} = \rho(\boldsymbol{A}_{j|i} \boldsymbol{x}_j^k - \boldsymbol{A}_{i|j} \boldsymbol{x}_i^{k+1}) - \boldsymbol{\lambda}_{j|i}^k \qquad \forall j \in \mathcal{V}_i \tag{8.13}$$

where $(\cdot)_+$ is projection onto the nonnegative orthant, i.e., setting all negative elements to zero.

Finally, for each signal sample we require a collection phase where the noisy observations from throughout the network are weighted and summed to produce our beamformed output signal. At each time sample t we therefore must perform sharing and averaging iterations many times for information to mix throughout the network. However, as we will show in our simulation results, information at distant nodes within the network often provide negligible performance gains. For the sake of our sparse PDMM beamformer we simply transmit the weighted observations of our active nodes to a single collection node (designated as the node closest to the source).

8.1.3 Numerical Simulation

We simulated a network with $N = 50$ microphone nodes and a source signal randomly placed in a 100 m $\times 100$ m $\times 100$ m free-space environment. The results are averaged over 20 realizations. The distances from node i to all other nodes were assumed to be known and neighbours were truncated to fall within a transmission distance of 50 m. The acoustic transfer function for each node was generated using the free-space model. The signal of interest was a 20 s speech sample randomly chosen from a 60 s recording. The interference was a randomly placed zero-mean Gaussian point source with power equal to -5 dB, 0 dB, and 5 dB when compared to the source signal. Estimation of the partial covariance matrix was assumed beforehand. The sample rate at each node was $f_s = 16$ kHz and processing was carried out on 25 ms Hann windowed blocks with a 50% overlap. The weight vector optimization process was performed asynchronously.

Figure 8.1 shows the effect of sparsity on our output SNR and on the communication power required for beamformer outputs per sample, when compared with the fully active MVDR node set (which uses distributed averaging for sample output). Due to our local covariance matrix approximation we see a drop of around 3 dB from the optimal case when all nodes are active, with a loss of at most 7 dB as our active node set falls to below 10 nodes. Communication power required for output (computed using [135] for only power amplifier transmission with free space parameters, i.e., a power cost simply proportional to the square of the distance between transmitter and receiver) in our sparse system using single node collection is initially higher than distributed averaging due to the expense of long range wireless communication with distant active nodes. However, our sparse system rapidly becomes more efficient as these distant (low fidelity nodes) are excluded, falling to around 15 % of the averaging power.

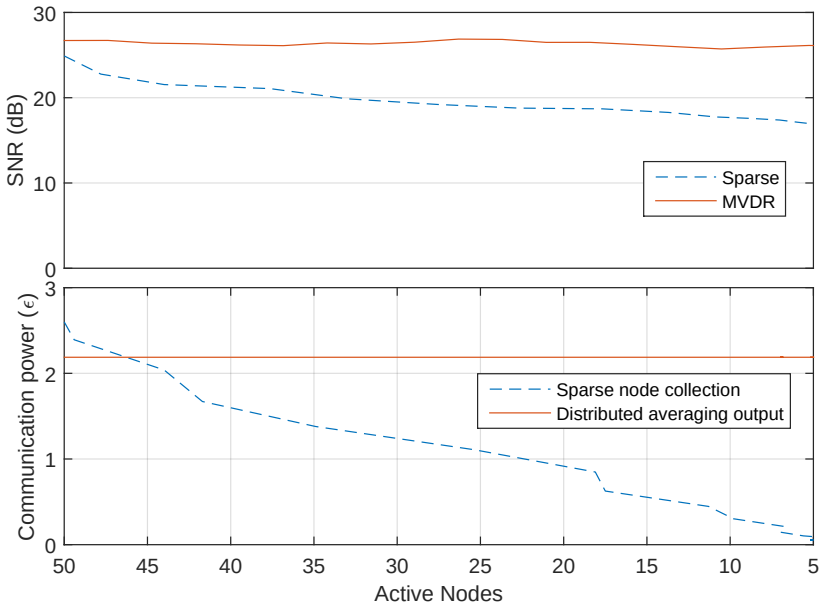


Figure 8.1: Top: SNR as a function of active nodes; Bottom: Communication Power as a function of active nodes.

8.1.4 Summary

We have motived, designed and tested a sparse distributed beamforming system that trades SNR performance for reduced inter-node power consumption. The l_1 norm is a natural first application of regularization to the problem due to the physical fall-off of sound power over distance and the resulting node fidelity pattern, particularly for large and sparse networks. Our system optimization performs updates that are asynchronous, independent, and do not rely on global collection phases. Further work would include research into more appropriate regularization functions that, for example, explicitly take into account the fidelity of each node and their distance from the source.

8.2 Application to Distributed Image Fusion

Image fusion is a signal processing method of combining multiple images of a common scene into a single image by exploiting the relevant information from these images [137, 101, 136], similar to the concept of diversity in multiple sensor electromagnetic communications systems. Arrays of imaging sensors, often distributed nonuniformly, may be used to produce varying images of a common scene that is then 'fused' at a central processing unit to produce a single image of higher fidelity than any of the constituent images [105, 52]. The imaging techniques used at each sensor may also vary, allowing for the fusion of data that would otherwise not be available to a single sensor. A common example of this is the fusion of high dimensional panchromatic images (for geometric detail) and low dimensional multispectral data (for colour information) in satellite imaging [122].

In recent years, two trends have arisen that motivate the concept of image fusion performed in a distributed manner. Firstly, advancements in wireless sensor network (WSN) technology [37, 72, 81] have meant that

small and low power sensor units equipped with microprocessors, wireless communications systems, and imaging sensors are becoming affordable for distributed surveillance [70, 63] or virtual reality mapping of real world environments [51]. These may be used to form adhoc WSNs with random topologies, such as aerial surveillance drone 'flocks' [74] that fuse images of the terrain below their flight paths at each time sample. In this case, sensor node energy consumption may be reduced by not requiring raw image transmission from each sensor to a central collector. Local communication between close neighbours may instead be used for the image fusion process across the network. The fused image may then be transmitted from a node if it is tapped for output.

The second motivation for distributed image fusion is the vast amount of data present in systems such as large radio telescope arrays [44], e.g., the proposed Square Kilometre Array. The high resolution of the images at each radio telescope coupled with the number of such telescopes make the transmission of raw data and the processing of fused images prohibitive in terms of communication bandwidth, storage capacity, computation power, and energy consumption [69]. Distributed fusion in this case will act to alleviate these issues through in-network local fusion iterations rather than transmission to, and processing at, a central collector.

While there has been significant recent progress in distributed data fusion techniques [19, 60] there has been little work in extending traditional image fusion to a distributed setting, suitable for fully decentralized processing. The work in [79] partially distributes the computation of fusion filters but still requires a central collection phase for final image estimation. For large WSNs or high volume systems such as radio telescope arrays this still results in an expensive bottleneck where image data from many sensors needs to be routed to a fusion centre.

In the case of a WSN image fusion with many partially overlapping fields of view (FOV), perhaps covering many square kilometers of total imaging area, we encounter an obvious system limitation: for a fully fused

image to be formed at a single sensor node there must be enough memory at said node to store, as well as enough communication power to transmit, this large image throughout the network. This clearly infeasible approach is also the solution that would arise from standard distributed consensus optimization techniques [15] where a common network-wide solution vector is optimized for. Instead, we will discuss a method that fuses redundant overlapping imaging areas while still maintaining the original image dimensions at each sensor.

We propose a new processing architecture for distributed image fusion (which will work in both the WSN and antenna array examples discussed above) based on elementwise general form consensus [15] using the Primal-Dual Method of Multipliers (PDMM) [155, 158]. Our system will optimize the image held by each sensor node by exploiting the additional information held by neighbouring nodes, effectively performing fusion on the mutual area viewed by each pair of nodes. Our algorithm will, in fact, exploit information from all nodes sharing mutual image areas provided the nodes that make up the communication path between any two nodes also share the same mutual image area. This process will operate using asynchronous and independent updates at each node, eliminating the need for a synchronous update iteration clock. The result of our algorithm is a network of nodes whose image observations may be improved via image information fused throughout the entire network using only local updates, without each node being required to contain a full image of the full network imaging area.

Section 8.2.1 will give an overview of the proposed system; section 8.2.2 will discuss the centralized TV-L1 algorithm; section 8.2.3 will present the derivation of our distributed PDMM fusion algorithm; section 8.2.5 will analyse the distributed PDMM algorithm; section 8.2.6 will present simulated results of our algorithm's performance; finally, section 8.2.7 will summarise the conclusions of the section.

8.2.1 System overview

We first outline the system we are proposing to better motivate the distributed image fusion algorithm discussed in section 8.2.3. Throughout the remainer of the section we will use the example of a network of aerial imaging drones (perhaps used for security surveillance or for geographical mapping) where each drone node is equipped with a single imaging sensor, a microprocessor, and electromagnetic (EM) communications for inter-node communication. These nodes form a time varying WSN $\mathcal{V}$ with cardinality $V = |\mathcal{V}|$, where communication is restricted to nodes connected by the edge set $\mathcal{E}$ with cardinality $E = |\mathcal{E}|$. That is, if two nodes i and j are within communication range of each other we say that the pair $(i, j) \in \mathcal{E}$.

Each node i records an image at time t, denoted $\boldsymbol{Z}_i(t)$, of the terrain directly below. For the sake of simplicity we assume that: the network topology remains fixed for the duration of each image fusion process and that the communication range of each sensor is low enough that the FOV of neighbouring nodes partially overlap. Additionally, for the purposes of our algorithm presentation we will not fuse single node image tracks over time. The advantage of multiple images taken at a single time sample are that time varying phenomena (such as vehicles for drone surveillance or oscillating pulsars for radio astronomy) will be captured rather than filtered out of a single image track as noise. We also therefore omit the time index t and assume images are processed on the same time sample (although our system could be easily extended to multiple time samples).

Once each node has captured the local imaging area (LIA) they each perform an image perspective transformation, in order for all images to share a common orthographic aerial view [52, 1]. Each node is then able to estimate the LIA overlap between neighbouring nodes by downsampling, transmitting, and comparing their images [27, 62]. At this point the distributed image fusion process may begin. Our distributed image fusion process is then able to exploit the overlap between a node i and most other nodes in the network by performing asynchronous optimiza-

tion updates at each node, as will be discussed in the following sections. At convergence the fused image held by each node i is of higher fidelity than the original image, having gained the information of overlapping images throughout the network.

Image outputs may now be drawn from the network in a number of different ways. Firstly, we may 'tap' a single node to view the fused image held by only this node. This would be an efficient way of drawing a high fidelity fused image of a node's LIA. Secondly, assuming each node is able to be tracked (e.g., via GPS) by an outside system operator it would be possible to determine a sparse set of nodes that would represent the fewest node FOVs required to cover the whole of the global imaging are (GIA). This sparse set of nodes (potentially an order of magnitude less than the total network size) would then each transmit the fused image of their LIA out of the network [69].

8.2.2 Centralized TV-L1 Image Fusion

Centralized image fusion has received significant attention over the past three decades and a number of competing methods exist that have been used with great success, with satellite [22] and medical [126] image fusion being two of the most prominent current application areas. Recently, the method of total variation (TV) denoising [147, 144] (and extensions involving wavelet regularization [74]) has seen a resurgence in popularity due to its relatively simple description and solution under the framework of convex optimization. The approach of TV-L1 denoising, in particular, has been used as a framework for efficient image fusion algorithms [147].

Our model begins by defining our discrete GIA image domain $\mathcal{I}$ as a regular grid with dimensions $H \times W$ where $\mathcal{I} = \{(p, q) \mid 1 \leq p \leq H,\ 1 \leq q \leq W\}$ with image observation $\boldsymbol{U} \in \mathbb{R}^{H \times W}$. The TV-L1 optimization criteria for a single image is then the combination of a total variation term of the denoised image $\boldsymbol{Z} \in \mathbb{R}^{H \times W}$ and a data error term between each

denoised image pixel $[\boldsymbol{Z}]_{p,q}$ and the observed image pixel $[\boldsymbol{U}]_{p,q}$, where $[\boldsymbol{Z}]_{p,q}$ represents the scalar element of $\boldsymbol{Z}$ at index (p, q). We may represent this as the unconstrained regularized optimization problem [74]

$$\text{minimize} \quad \alpha \left\| \nabla \boldsymbol{Z} \right\|_1 + \sum_{(p,q)\in\mathcal{I}} \left| [\boldsymbol{Z}]_{p,q} - [\boldsymbol{U}]_{p,q} \right|, \tag{8.14}$$

where ∇ is the anisotropic discrete derivative operator, and α is a tuneable parameter set by the implementer that determines the importance of total image variation relative to observed pixel reconstruction accuracy. When fusing N observed images that all cover the same global set of pixels $(\boldsymbol{U}_1, \ldots, \boldsymbol{U}_V) \in \mathbb{R}^{H\times W}$, we may extend the data error term to include these observations [74], i.e.,

$$\text{minimize} \quad \sum_{i\in\mathcal{V}} \alpha \left\| \nabla \boldsymbol{Z} \right\|_1 + \sum_{i\in\mathcal{V}} \sum_{(p,q)\in\mathcal{I}} \left| [\boldsymbol{Z}]_{p,q} - [\boldsymbol{U}_i]_{p,q} \right|, \tag{8.15}$$

where the summation over the total variation term simply scales it by V, retaining the relative effect of the tuning parameter α. Optimizing this function fuses the information of multiple images of a common global pixel set, improving the fidelity of the final image. In contrast, traditional centralized methods of performing this optimization assume that all image data is sent to a central processor for computation, resulting in expensive communication costs for networks that cover a large area. Additionally, in high data volume applications it may be physically impossible to store the images from all sensors in a single location due to memory limitations.

8.2.3 Distributed PDMM Image Fusion

To overcome the problem of long distance data transmission, central data housing, and central computation, we will distribute the image fusion described by (8.15) among the V sensor nodes of the network. We begin by presenting the most obvious method of distributing the fusion of image

data - the popular method of distributed consensus optimization [15]. We introduce a local copy of the global optimization variable $\boldsymbol{Z}$ at each node i denoted $\boldsymbol{Z}_i$, and apply edge-wise equality constraints between each pair of neighbouring nodes. Assuming the network is connected (i.e. it is possible to traverse the network from a given node to any other), these constraints will ensure all copies are equal and we may construct an equivalent problem to the centralized problem (8.15), i.e.

$$\begin{aligned} &\text{minimize} \quad \sum_{k\in\mathcal{V}} \left(\alpha_i \left\| \nabla \boldsymbol{Z}_i \right\|_1 + \sum_{(p,q)\in\mathcal{I}} \left| [\boldsymbol{Z}_i]_{p,q} - [\boldsymbol{U}_i]_{p,q} \right| \right) \\ &\text{subject to} \quad [\boldsymbol{Z}_i]_{p,q} = [\boldsymbol{Z}_j]_{p,q} \quad \forall (p,q) \in \mathcal{I},\ \forall (i,j) \in \mathcal{E}, \end{aligned} \tag{8.16}$$

where $\alpha_i = \alpha \ \forall i \in \mathcal{V}$. Solving this problem using PDMM or ADMM does not require a central fusion processor, and only relies on local data transmission between neighbouring nodes. There are still two important limitations to this approach: each node i is required to store a full copy of the GIA represented by the variable $\boldsymbol{Z}_i$; and during optimization of this problem we require transmission of this full GIA variable copy. In cases where the total GIA is small and all sensors have a high proportion of this area in their FOV (i.e., there is high redundancy of a relatively small imaging area), this approach is feasible. However, for our system to be scalable to applications involving GIAs that are orders of magnitude larger than the LIA of any single sensor with partial overlaps in FOV, we are unable to store and transmit these full GIA copies.

To solve the problem of GIA data storage and transmission, we extend the consensus approach of (8.16) to a type of general form consensus [158, 15] that enforces partial consensus of neighbouring nodes through elementwise equality constraints of subsets of variable components. We refer to this as general form neighbouring consensus (GFNC). In this way, we are able to retain the original LIA image size at each node while optimizing for the redundancy in overlapping regions. We begin by allowing each node i to view a new LIA image domain with independent dimen-

sions $H_i \times W_i$ denoted $\mathcal{I}_i$ such that each LIA image domain is a subset of the GIA image domain, i.e., $\mathcal{I}_i \subset \mathcal{I}$ for all i. We then denote our new LIA observation as $\boldsymbol{U}_i \in \mathbb{R}^{H_i \times W_i}$ and our new local optimization variable as $\boldsymbol{X}_i \in \mathbb{R}^{H_i \times W_i}$. We may now pose the partial consensus optimization problem

$$\begin{aligned} \text{minimize} \quad & \sum_{k \in \mathcal{V}} \left(\alpha_i \left\| \nabla \boldsymbol{X}_i \right\|_1 + \sum_{(p,q) \in \mathcal{I}_i} \left| [\boldsymbol{X}_i]_{p,q} - [\boldsymbol{U}_i]_{p,q} \right| \right) \\ \text{subject to} \quad & [\boldsymbol{X}_i]_{p,q} = [\boldsymbol{X}_j]_{p,q} \; \forall (p,q) \in \mathcal{I}_i \cap \mathcal{I}_j, \; \forall (i,j) \in \mathcal{E}, \end{aligned} \tag{8.17}$$

where we now only enforce consensus between pixels that are shared in the LIA's of neighbouring nodes.

To pose our new partial consensus problem in a form more readily solvable by convex optimization methods we will vectorize our variables $\boldsymbol{x}_i = \mathrm{vec}(\boldsymbol{X}_i)$ and $\boldsymbol{u}_i = \mathrm{vec}(\boldsymbol{U}_i)$ so that $\{\boldsymbol{x}_i, \boldsymbol{u}_i\} \in \mathbb{R}^{H_i W_i}$ with new indices defined as $g_i = (H_i(q-1) + p)$. We may then write our optimization as

$$\begin{aligned} \text{minimize} \quad & \sum_{k \in \mathcal{V}} \left(\alpha_i \left\| \boldsymbol{D}_i \boldsymbol{x}_i \right\|_1 + \left\| \boldsymbol{x}_i - \boldsymbol{u}_i \right\|_1 \right) \\ \text{subject to} \quad & \boldsymbol{A}_{i|j} \boldsymbol{x}_i = \boldsymbol{A}_{j|i} \boldsymbol{x}_l \quad \forall (i,j) \in \mathcal{E}, \end{aligned} \tag{8.18}$$

where $\boldsymbol{D}_i$ is the derivative operator for vector k defined by the relation $D_i \mathrm{vec}(\cdot) = \nabla(\cdot)$, and the constraint matrices $\boldsymbol{A}_{i|j}$ and $\boldsymbol{A}_{j|i}$ may be seen as relative alignment matrices for the edge (i,j) that contain all zeros apart from at most a single entry of 1 for each row. These would be constructed from the alignment process carried out prior to image fusion, and would effectively select and permute entries in order to compare overlapping pixel areas from the images held by nodes i and j. The equivalence of the distributed partial consensus problem (8.18) and the centralized problem (8.15) will be discussed further in section 8.2.5.

We may now use the primal-dual method of multipliers [158, 156] to perform distributed, independent, and asynchronous updates across the

network using the local primal and dual variable updates

$$\boldsymbol{x}_i^{k+1} = \arg\min_{\boldsymbol{x}_i} \Bigg[\alpha_i \big\| \boldsymbol{D}_i \boldsymbol{x}_i \big\|_1 + \big\| \boldsymbol{x}_i - \boldsymbol{u}_i \big\|_1 + \boldsymbol{x}_i^T \Big(\sum_{l \in \mathcal{V}_i} \boldsymbol{A}_{i|j}^T \boldsymbol{\lambda}_{j|i}^k \Big) + \sum_{l \in \mathcal{V}_i} \frac{\rho}{2} \| \boldsymbol{A}_{i|j} \boldsymbol{x}_i - \boldsymbol{A}_{j|i} \boldsymbol{x}_l^k \|_2^2 \Bigg] \quad \forall i \in \mathcal{U}, \tag{8.19}$$

$$\boldsymbol{\lambda}_{i|j}^{k+1} = \rho(\boldsymbol{A}_{j|i} \boldsymbol{x}_l^k - \boldsymbol{A}_{i|j} \boldsymbol{x}_i^{k+1}) - \boldsymbol{\lambda}_{j|i}^i \quad \forall i \in \mathcal{V},\ \forall l \in \mathcal{V}_i. \tag{8.20}$$

In order to pose our problem in a form solvable by many common optimization methods, we combine the l_1 terms and expand out the quadratic edge-wise penalty to describe the primal update step as the optimal point $\boldsymbol{z}_i^*$ of the convex problem

$$\text{minimize} \quad \frac{1}{2} \boldsymbol{x}_i^T \boldsymbol{R}_i \boldsymbol{x}_i + \boldsymbol{w}_i^T \boldsymbol{x}_i + \| \boldsymbol{M}_i \boldsymbol{x}_i - \boldsymbol{q}_i \|_1, \tag{8.21}$$

where

$$\boldsymbol{R}_i = \rho \sum_{l \in \mathcal{V}_i} \boldsymbol{A}_{i|j}^T \boldsymbol{A}_{i|j}, \tag{8.22}$$

$$\boldsymbol{w}_i = \sum_{l \in \mathcal{V}_i} \left(\boldsymbol{\lambda}_{j|i}^{iT} \boldsymbol{A}_{i|j} - \rho \boldsymbol{x}_l^T \boldsymbol{A}_{j|i}^T \boldsymbol{A}_{i|j} \right), \tag{8.23}$$

$$\boldsymbol{M}_i = [\alpha_i \boldsymbol{D}_i \quad \boldsymbol{I}]^T, \qquad \boldsymbol{q}_i = [\boldsymbol{0} \quad \boldsymbol{u}_i]^T, \tag{8.24}$$

$\boldsymbol{0}$ is the zero vector, and $\boldsymbol{I}$ is the identity matrix. We may equivalently pose problem (8.21) as the quadratic program

$$\begin{array}{ll} \text{minimize} & \frac{1}{2} \boldsymbol{x}_i^T \boldsymbol{R}_i \boldsymbol{x}_i + \boldsymbol{w}_i^T \boldsymbol{x}_i + \boldsymbol{1}^T \boldsymbol{y}_i \\ \text{subject to} & \boldsymbol{y}_i \geq \boldsymbol{M}_i \boldsymbol{x}_i - \boldsymbol{q}_i \\ & \boldsymbol{y}_i \geq -\boldsymbol{M}_i \boldsymbol{x}_i + \boldsymbol{q}_i, \end{array} \tag{8.25}$$

where the two constraints and the inner product $\boldsymbol{1}^T \boldsymbol{y}_i$ ensure that the l_1 norm of the quantity $\boldsymbol{M}_i \boldsymbol{x}_i - \boldsymbol{q}_i$ is taken at convergence.

8.2.4 Improving Computational Efficiency with FS-PDMM

In this subsection we will discuss the application of function splitting to the image fusion problem to improve local computational efficiency. The quadratic program (8.25) may be simple to solve for very small images where the dimensionality of our optimization variables $\boldsymbol{z}_i$ low, but for most practical imaging applications with images in the megapixel range this optimization becomes prohibitively slow. To address this, one approach explored in [74] was to reduce optimization dimensionality of the central problem (8.15) through wavelet transformation [3]. The optimization procedure was then carried out in the lower dimensionality wavelet domain rather than in the very high dimensional pixel domain. This is effective in central applications, but for distributed problems where computational power is even more constrained we would benefit from further simplification of the optimization problem.

The FS-PDMM algorithm in chapter 4 presents an asynchronous method of splitting coupled problems, such as common regularized optimization problems, between two mathematically separate nodes in order to reduce the computational cost and eliminate the overhead of extra optimization packages. To apply FS-PDMM to this problem we initially combine the two l_1 terms in problem (8.18) as in the previous section using (8.24). This function will now be treated as our regularization term, while our cost function term will be empty. This gives the problem

$$\begin{aligned} \text{minimize} \quad & \sum_{k\in\mathcal{V}} \left(f_i(\boldsymbol{x}_i) + R_i(\boldsymbol{x}_i)\right) \\ \text{subject to} \quad & \boldsymbol{A}_{i|j}\boldsymbol{x}_i = \boldsymbol{A}_{j|i}\boldsymbol{x}_l \quad \forall (i,j) \in \mathcal{E}, \end{aligned} \tag{8.26}$$

where $f_i(\boldsymbol{x}_i) = 0$ and $R_i(\boldsymbol{x}_i) = \|\boldsymbol{M}_i\boldsymbol{x}_i - \boldsymbol{q}_i\|_1$. Compared with the FS-PDMM algorithm, we note that there is an extra constant term $\boldsymbol{q}_i$ in the regularization function R_i. This requires a minor adjustment to the FS-PDMM update equations to include this constant, but the general algorithm remains the same. We therefore have the following FS-PDMM up-

date iterations for the image fusion problem

$$\boldsymbol{x}_i^{k+1} = \left(\rho \boldsymbol{M}_i^T \boldsymbol{M}_i + \rho \sum_{j \in \mathcal{V}_i} \boldsymbol{A}_{i|j}^T \boldsymbol{A}_{i|j}\right)^{-1} \left(\boldsymbol{M}_i^T (\rho \boldsymbol{z}_i^k + \rho \boldsymbol{q}_i - \boldsymbol{\nu}_i^k) \right. \tag{8.27}$$

$$\left. - \sum_{j \in \mathcal{V}_i} \boldsymbol{A}_{i|j}^T (\boldsymbol{\lambda}_{ji}^k + \rho \boldsymbol{A}_{j|i} \boldsymbol{x}_j^k)\right) \tag{8.28}$$

$$\boldsymbol{z}_i^{k+1} = \left(2\boldsymbol{M}_i \boldsymbol{x}_i^{k+1} - 2\boldsymbol{q}_i - \boldsymbol{z}_i^k + \boldsymbol{\nu}_i^k/\rho - \alpha_i/\rho\right)_+ - \left(-2\boldsymbol{M}_i \boldsymbol{x}_i^{k+1} + 2\boldsymbol{q}_i + \boldsymbol{z}_i^k - \boldsymbol{\nu}_i^k/\rho - \alpha_i/\rho\right)_+ \tag{8.29}$$

$$\boldsymbol{\nu}_i^{k+1} = \rho(2\boldsymbol{M}_i \boldsymbol{x}_i^{k+1} - 2\boldsymbol{q}_i - \boldsymbol{z}_i^{k+1} - \boldsymbol{z}_i^k) + \boldsymbol{\nu}_i^k \tag{8.30}$$

$$\boldsymbol{\lambda}_{i|j}^{k+1} = \rho(\boldsymbol{A}_{j|i} \boldsymbol{x}_j^k - \boldsymbol{A}_{i|j} \boldsymbol{x}_i^{k+1}) - \boldsymbol{\lambda}_{j|i}^k \qquad \forall j \in \mathcal{V}_i \tag{8.31}$$

for all $i \in \mathcal{V}$. We now have four simple updates to perform each iteration k. The $\boldsymbol{x}$ update requires the solution of a linear system, where the matrix to be inverted is constant for all iterations and may be possible to cache. The $\boldsymbol{z}$ update is the soft thresholding operator which is efficient to evaluate, and the final two dual variable updates are simple linear transformations. These four updates, particularly for high dimensional problems, are far simpler to solve than the quadratic program proposed in (8.25).

8.2.5 Equivalence to central fusion

In this section we show that our distributed algorithm is equivalent to centrally fusing the overlapping regions of all images. We assume that the edge-wise consensus constraints in (8.17) are met and additionally that any two nodes sharing a common subset of pixels are connected by other nodes that also share this same pixel subset to ensure that these pixels all reach consensus (analysis of the extent of this effect is worth pursuing in future work, but our simulated experimental data suggests that the number of pixels not satisfying this second assumption are neglible). This analysis is intended as a sanity check, to validate that the proposed approach solves our original optimization problem.

We define new disjoint image subsets $(\mathcal{A}_1, \ldots, \mathcal{A}_M) \in \mathcal{I}$ that represent overlapping areas with fixed node sets. Geometrically these subsets are exactly the polygons that are formed from the overlapping LIA borders, and will be referred to as fixed observer-set polygons (FOSP). The union of these FOSPs therefore cover the same area as the union of the LIAs, i.e., $(\mathcal{A}_1 \cup \cdots \cup \mathcal{A}_M) = (\mathcal{I}_1 \cup \cdots \cup \mathcal{I}_N)$. Expanding out the total variation term in equation (8.17) gives

$$\sum_{i\in\mathcal{V}} \Big(\sum_{(p,q)\in\mathcal{I}_i} \alpha_i \big(|[\boldsymbol{X}_i]_{p+1,q} - [\boldsymbol{X}_i]_{p,q}| + |[\boldsymbol{X}_i]_{p,q+1} - [\boldsymbol{X}_i]_{p,q}| \big) + \sum_{(p,q)\in\mathcal{I}_i} \big| [\boldsymbol{X}_i]_{p,q} - [\boldsymbol{U}_i]_{p,q} \big| \Big) \tag{8.32}$$

$$= \sum_{m=1}^{M} \sum_{(p,q)\in\mathcal{A}_m} \sum_{i\in\mathcal{V}_m} \alpha_i \big(|[\boldsymbol{X}_i]_{p+1,q} - [\boldsymbol{X}_i]_{p,q}| + |[\boldsymbol{X}_i]_{p,q+1} - [\boldsymbol{X}_i]_{p,q}| \big) + \big| [\boldsymbol{X}_i]_{p,q} - [\boldsymbol{U}_i]_{p,q} \big| \tag{8.33}$$

$$= \sum_{m=1}^{M} \Bigg(\sum_{i\in\mathcal{V}_m} \Big(\sum_{(p,q)\in\mathcal{A}_m} \alpha_i \big(|\boldsymbol{X}_{p+1,q} - \boldsymbol{Z}_{p,q}| + |\boldsymbol{X}_{p,q+1} - \boldsymbol{X}_{p,q}| \big) \Big) + \sum_{i\in\mathcal{V}_m} \sum_{(p,q)\in\mathcal{A}_m} \big| [\boldsymbol{X}]_{p,q} - [\boldsymbol{U}_i]_{p,q} \big| \Bigg), \tag{8.34}$$

where the node set $\mathcal{V}_m = \{k | \mathcal{A}_m \cap \mathcal{I}_i \neq \varnothing\}$ denotes the set of all nodes contributing to FOSP m, the three summations in equation (8.33) denote summation over all FOSPs, summation over all pixels within each FOSP, and summation over all nodes contributing to each FOSP, respectively, and the i indices previously present on the $\boldsymbol{Z}$ variables have been omitted in equation (8.34) since the consensus constraints constrain all copies of these values to equality.

By inspection we see that equation (8.34) is a summation of M centralized TV-L1 fusion optimizations, each equivalent to problem (8.15) carried out on an 'overlap' image subset. In other words, the distributed partial

consensus optimization presented in problem (8.17) effectively fuses the pixels of all FOSPs, even when nodes contributing to these regions are not all within the same neighbourhood.

8.2.6 Numerical Simulation

A random ad-hoc network was placed on a 800x800 pixel global imaging area (©Institute of Geodesy and Photogrammetry, ETH Zurich), each with a local imaging area size of 256x256 pixels and a communication range of 100 pixels. Pixel values were in the continuous range 0 to 1 for the purpose of optimization. The received images at each node were corrupted by zero mean independent Gaussian noise with standard deviation of 0.003 as well as 0.05 density salt and pepper noise [74]. Perspective mapping and alignment procedures were assumed to have been carried out prior to the image fusion phase, resulting in the alignment matrices required by problem (8.18). The PDMM algorithm was run by randomly triggering a node at each iteration for update for an average of 10 iterations per node.

The peak signal-to-noise ratio (PSNR) was computed for a 512x512 area in the centre of the network so as to avoid areas on the edge of the network with no redundancy, with only the fewest fused node images required to cover the area used for PDMM network output. This process was repeated for 10 random instances of the network over a range of network node sizes. This was compared to an image of the same area resulting from a stitched collection of single images denoised using TV-L1 with no redundancy (single image denoising), as well as this area fused centrally with maximum redundancy. The tuning parameter λ was varied to find the optimal trade-off value based on the PSNR performance.

We begin with a pictorial display of the performance of the 50 node PDMM fusion algorithm in figure 8.2, comparing the raw noisy GIA and LIA, the locally (single image) denoised GIA and LIA, and the PDMM fused GIA and LIA. The local denoising does well to remove most noise

at the expense of fine details, whereas the redundancy exploited by the PDMM fusion process allows fine details to be retained while reducing the noise present. The effect of noise on the PDMM fused GIA is more prevalent at the outer edges of the GIA. This is caused by a lack of redundancy since these areas are only viewable from one or two nodes, resulting in performance similar to the locally denoised pixels of these outer areas.

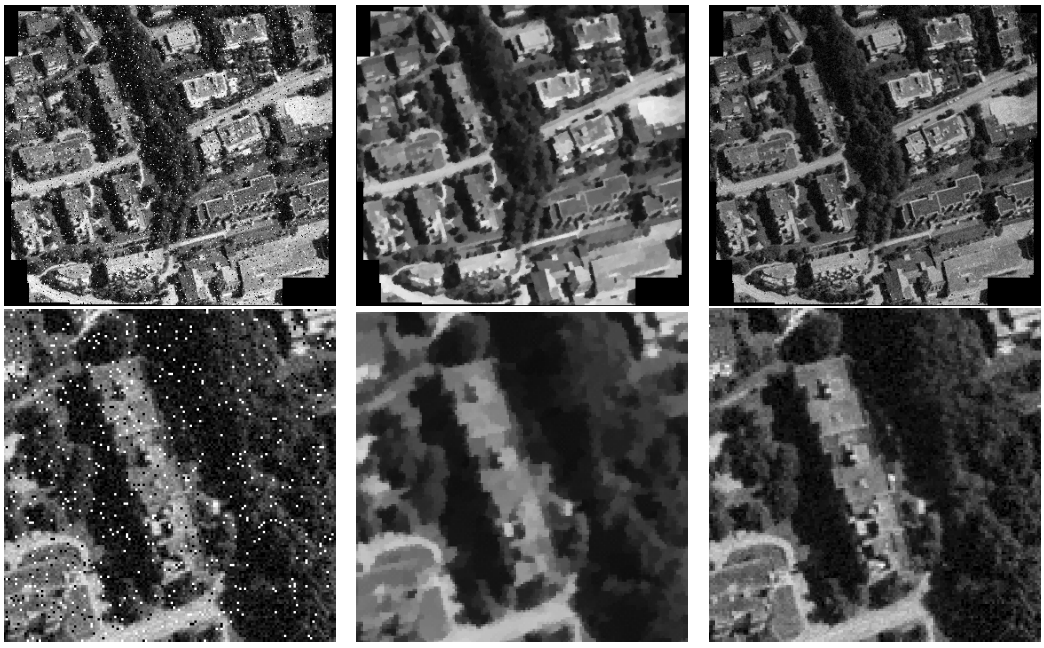


Figure 8.2: Top row from left: noisy GIA; locally denoised GIA; PDMM denoised GIA. Bottom row from left: noisy LIA; locally denoised LIA; PDMM denoised LIA.

Figure 8.3 shows the statistical performance of the algorithm averaged over 10 instances with cross-validated tuning parameter. The PSNR of the PDMM algorithm, the central fusion process, and local independent denoising are compared as a function of average nodes per pixel and contrasted with the energy consumption ([135] using free space parameters) of these systems with a central collector base station 1 km away. We see that the performance of our distributed algorithm is roughly equal to that

of the central fusion process while consuming significantly less energy since redundant information is fused within the network prior to transmission to the base station, rather than each node transmitting a raw image observation.

Finally, figure 8.4a and 8.4a show the global convergence speed and runtime, respectively, of the distributed image fusion when comparing conventional PDMM to FS-PDMM. Instances where conventional PDMM failed to converge on a solution were removed in order to compare the best case of PDMM with FS-PDMM. We see a reduction in runtime of FS-PDMM by around two orders of magnitude as compared with conventional PDMM, which uses local ADMM splitting to solve subproblems. The global convergence speed is slightly reduced by using FS-PDMM.

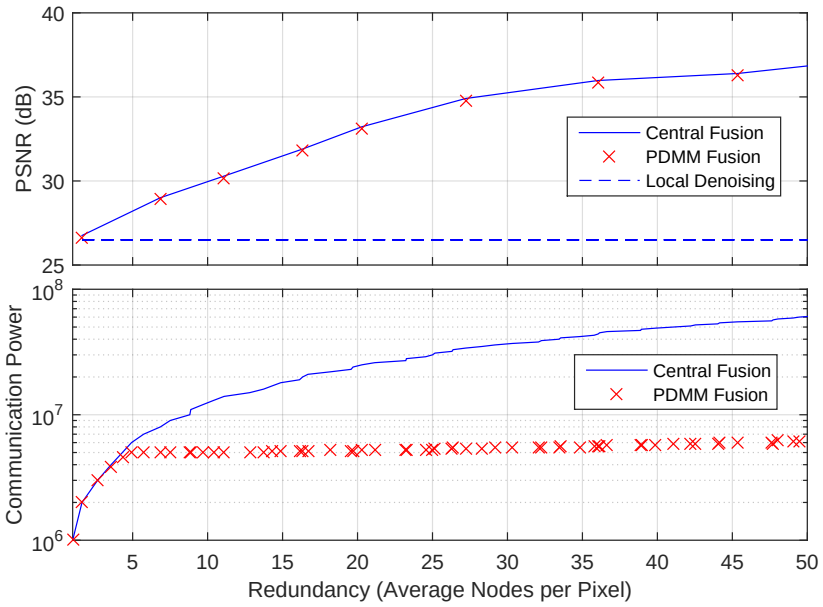
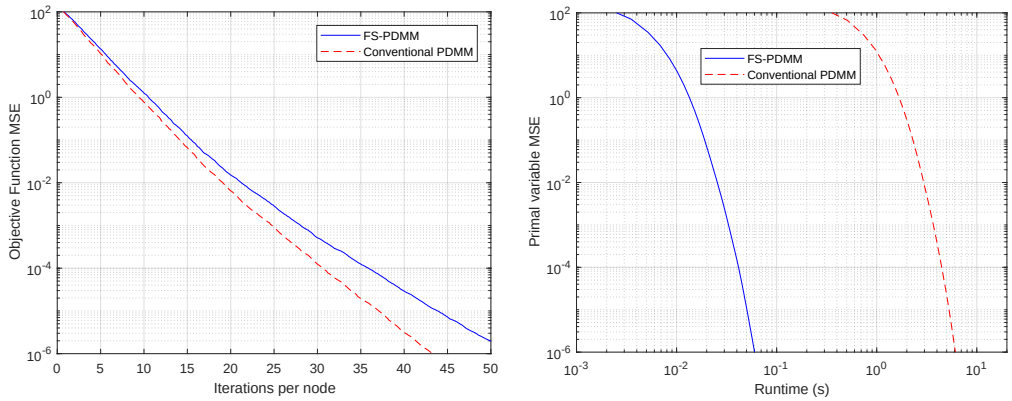


Figure 8.3: PSNR and communication power versus nodes per pixel.



(a) Convergence rate of conventional PDMM and FS-PDMM applied to distributed image fusion

(b) Computational efficiency of conventional PDMM and FS-PDMM applied to distributed image fusion

Figure 8.4: Performance of FS-PDMM and conventional PDMM.

8.2.7 Summary

A fully asynchronous distributed image fusion system was developed for a general network of imaging sensors with partially overlapping FOVs. We show that performing image fusion in this manner is equivalent to performing centralized fusion over each overlapping FOV, even when non-neighbouring nodes view common overlaps. Simulated results show that we achieve roughly equal performance to the centralized case while conserving considerable transmission energy. We also show that applying the FS-PDMM algorithm to the image fusion problem leads to a runtime reduction of around two orders of magnitude.

Chapter 9

Conclusions and Future Works

9.1 Conclusions

This thesis discussed recent developments in the area of distributed wireless sensor network processing and has specifically dealt with the distributed optimization algorithm of PDMM. We have developed the FS-PDMM, QA-PDMM, FSQA-PDMM, and FT-PDMM algorithms, all of which are variations on conventional PDMM. These variations were motivated by the necessity for efficient and simple local computations when performing signal processing in wireless sensor networks. Distributed acoustic beamforming served as a motivating application example for the simplification of PDMM, and was used along with distributed image fusion as application examples for practical distributed signal processing algorithms.

The FS-PDMM and QA-PDMM algorithms were developed to improve the computational efficiency of regularized optimization and smooth cost function optimization, respectively. Additionally, these methods eliminate the need for numerical optimization packages for many common problems. These algorithms were analysed to have a convergence rate of $O(1/k)$ for closed, proper, and convex functions. Simulated experiments on elastic net regularized least squares optimization, for FS-PDMM, and logistic regression, for QA-PDMM, showed that roughly an order of mag-

nitude improvement in computational efficiency was possible. The FSQA-PDMM algorithm was then developed, analysed, and simulated, combining the benefits of both previous methods and allowing for efficient optimization when performing distributed optimization with a smooth cost function and non-smooth regularization function. The FT-PDMM was developed as the primal variation on PDMM, and was experimentally shown to converge in finite iterations when optimizing quadratic consensus problems.

Distributed signal processing applications of acoustic beamforming and image fusions were demonstrated, using FS-PDMM to reduce the computational complexity of local updates. The beamformer included a novel sparsity approach that naturally selected the most useful sensor contributions, allowing less relevant nodes to save communication energy by not transmitting signals for beamformer combination. The image fusion algorithm allowed for a network imaging sensors with partially overlapping fields of view to fuse their common imaging areas and improve image fidelity in-network. This greatly reduced the potentially high cost of transmitting redundant raw images to a central fusion center.

9.2 Future Works

As future work, we intend to apply the algorithms developed in this thesis to more practical problems in the areas of distributed signal processing and distributed machine learning with the eventual real-world implementation of these methods in a lab-based sensor network. We would like to further analyse the quadratic parameter matrix in QA-PDMM in order to determine an optimal setting for a given problem, with low computational complexity. Additionally, we would like to further explore finite-time convergence and see whether more general cost functions could benefit from the methods used in FT-PDMM.

Appendix A

Appendix

A.1 QA-PDMM Convergence Inequality 1

Recall the bounds on the Hessian assumed in equation (5.10). The lower bound implies that for all $i \in \mathcal{N}$, each local function f_i is strongly convex with constant q_i, i.e.,

$$(\nabla f_i(\boldsymbol{x}_i) - \nabla f_i(\boldsymbol{y}_i))^T(\boldsymbol{x}_i - \boldsymbol{y}_i) \geq q_i\|\boldsymbol{x}_i - \boldsymbol{y}_i\|_2^2 \,\forall \boldsymbol{x}_i, \boldsymbol{y}_i \in \mathbb{R}^{n_i}. \tag{A.1}$$

The upper bound is equivalent to Lipschitz continuous gradients of the local functions f_i for all $i \in \mathcal{N}$ with constant Q_i, i.e.,

$$\|\nabla f_i(\boldsymbol{x}_i) - \nabla f_i(\boldsymbol{y}_i)\|_2 \leq Q_i\|\boldsymbol{x}_i - \boldsymbol{y}_i\|_2 \,\forall \boldsymbol{x}_i, \boldsymbol{y}_i \in \mathbb{R}^{n_i}. \tag{A.2}$$

From equation (2.33), the unaugmented primal-dual Lagrangian of problem (2.32) is given by $L(\boldsymbol{x}, \boldsymbol{\lambda}) = L_\Phi(\boldsymbol{x}, \boldsymbol{\lambda}) - \Phi(\boldsymbol{x}, \boldsymbol{\lambda})$, which has the KKT conditions

$$\nabla f_i(\boldsymbol{x}_i^*) + \sum_{j \in \mathcal{V}_i} \boldsymbol{A}_{i|j}^T \boldsymbol{\lambda}_{j|i}^* = 0 \quad \forall i \in \mathcal{N} \tag{A.3}$$

$$\boldsymbol{A}_{i|j}\boldsymbol{x}_i^* - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^* = 0 \quad \forall i, j \in \mathcal{E} \tag{A.4}$$

$$\boldsymbol{\lambda}_{i|j}^* + \boldsymbol{\lambda}_{j|i}^* = 0 \quad \forall i, j \in \mathcal{E}, \tag{A.5}$$

where we have used $g_i = f_i \ \forall i \in \mathcal{V}$. The optimality condition for the quadratically approximated primal variable update $\boldsymbol{x}_i^{k+1}$ in equation (5.5) is

$$\begin{aligned}\nabla f_i(\boldsymbol{x}_i^k) + H_i^k(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k) + \sum_{j\in\mathcal{V}_i} \boldsymbol{A}_{i|j}^T \boldsymbol{\lambda}_{j|i}^k \\ + \sum_{j\in\mathcal{V}_i} \rho \boldsymbol{A}_{i|j}^T(\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1} - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^k) = 0. \end{aligned} \tag{A.6}$$

We define a new function $f_i^k(\boldsymbol{x}_i)$ at iteration k for node i as

$$\begin{aligned} f_i^k(\boldsymbol{x}_i) =& f_i(\boldsymbol{x}_i^k) + \langle \boldsymbol{x}_i - \boldsymbol{x}_i^k, \nabla f_i(\boldsymbol{x}_i^k)\rangle \\ &+ \frac{1}{2}\|\boldsymbol{x}_i - \boldsymbol{x}_i^k\|_{H_i^k}^2, \end{aligned}$$

where $f_i^k(\boldsymbol{x}_i)$ is simply a quadratic approximation of f_i at $\boldsymbol{x}_i^k$. Upon introducing $f_i^k(\boldsymbol{x}_i)$, (A.6) can be rewritten as

$$\begin{aligned} \nabla f_i^k(\boldsymbol{x}_i^{k+1}) =& \nabla f_i(\boldsymbol{x}_i^k) + H_i^k(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k) \\ =& \sum_{j\in\mathcal{V}_i} \boldsymbol{A}_{i|j}^T \boldsymbol{\lambda}_{i|j}^{k+1}. \end{aligned} \tag{A.7}$$

We now derive two different but equivalent expressions for the quantity $\sum_{i\in\mathcal{V}}(\nabla f_i^k(\boldsymbol{x}_i^{k+1}) - \nabla f_i(\boldsymbol{x}_i^\star))^T(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^*)$. The first expression is given by

$$\begin{aligned} &\sum_{i\in\mathcal{V}}(\nabla f_i^k(\boldsymbol{x}_i^{k+1}) - \nabla f_i(\boldsymbol{x}_i^\star))^T(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^*) \\ &= \sum_{i\in\mathcal{V}} \left(\sum_{j\in\mathcal{V}_i} \boldsymbol{A}_{i|j}^T \boldsymbol{\lambda}_{i|j}^{k+1}\right)^T (\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^*) \\ &\quad - \sum_{i\in\mathcal{V}} \left(\sum_{j\in\mathcal{V}_i} \boldsymbol{A}_{i|j}^T \boldsymbol{\lambda}_{i|j}^\star\right)^T (\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^*) \\ &= \sum_{i\in\mathcal{V}} \left(\sum_{j\in\mathcal{V}_i} \boldsymbol{A}_{i|j}^T(\rho(\boldsymbol{A}_{j|i}\boldsymbol{x}_j^k - \boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1}) - \boldsymbol{\lambda}_{j|i}^k)\right)^T (\boldsymbol{x}_i^{k+1} \end{aligned}$$

$$
\begin{aligned}
&- \boldsymbol{x}_i^*) - \sum_{i\in\mathcal{V}} \left(\sum_{j\in\mathcal{V}_i} \boldsymbol{A}_{i|j}^T \boldsymbol{\lambda}_{i|j}^\star \right)^T (\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^*) \\
&= \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{V}_i} \Big[\rho(\boldsymbol{A}_{j|i}\boldsymbol{x}_j^k - \boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1}) + \boldsymbol{\lambda}_{j|i}^{k+1} \\
&\quad - \boldsymbol{\lambda}_{j|i}^k \Big]^T \boldsymbol{A}_{i|j}(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^*) \\
&\quad - \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{V}_i} \boldsymbol{\lambda}_{i|j}^{\star,T} \boldsymbol{A}_{i|j}(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^*) \\
&\quad - \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{V}_i} \boldsymbol{\lambda}_{j|i}^{k+1} \boldsymbol{A}_{i|j}(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^*),
\end{aligned} \tag{A.8}
$$

and the second expression is given by

$$
\begin{aligned}
&\sum_{i\in\mathcal{V}} (\nabla f_i^k(\boldsymbol{x}_i^{k+1}) - \nabla f_i(\boldsymbol{x}_i^\star))^T (\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^*) \\
&= \sum_{i\in\mathcal{V}} \left(\sum_{j\in\mathcal{V}_i} \boldsymbol{A}_{i|j}^T (\boldsymbol{\lambda}_{i|j}^{k+1} - \boldsymbol{\lambda}_{i|j}^*) \right)^T (\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^*) \\
&= \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{V}_i} (\boldsymbol{\lambda}_{i|j}^{k+1} - \boldsymbol{\lambda}_{i|j}^*)^T \boldsymbol{A}_{i|j} \boldsymbol{x}_i^{k+1} \\
&\quad - \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{V}_i} (\boldsymbol{\lambda}_{i|j}^{k+1} - \boldsymbol{\lambda}_{i|j}^*)^T \boldsymbol{A}_{i|j} \boldsymbol{x}_i^* \\
&= \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{V}_i} \left[\boldsymbol{A}_{j|i}\boldsymbol{x}_j^k - \frac{1}{\rho}(\boldsymbol{\lambda}_{j|i}^k + \boldsymbol{\lambda}_{i|j}^{k+1}) \right]^T (\boldsymbol{\lambda}_{i|j}^{k+1} - \boldsymbol{\lambda}_{i|j}^\star) \\
&\quad - \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{V}_i} (\boldsymbol{\lambda}_{i|j}^{k+1} - \boldsymbol{\lambda}_{i|j}^*)^T \boldsymbol{A}_{i|j} \boldsymbol{x}_i^* \\
&= \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{V}_i} \left[\boldsymbol{A}_{j|i}(\boldsymbol{x}_j^k - \boldsymbol{x}_j^{k+1}) - \frac{1}{\rho}(\boldsymbol{\lambda}_{j|i}^k + \boldsymbol{\lambda}_{i|j}^{k+1}) \right]^T (\boldsymbol{\lambda}_{i|j}^{k+1} \\
&\quad - \boldsymbol{\lambda}_{i|j}^\star) + \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{V}_i} \boldsymbol{A}_{j|i} \boldsymbol{x}_j^{k+1,T} (\boldsymbol{\lambda}_{i|j}^{k+1} - \boldsymbol{\lambda}_{i|j}^\star) \\
&\quad - \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{V}_i} (\boldsymbol{\lambda}_{i|j}^{k+1} - \boldsymbol{\lambda}_{i|j}^*)^T \boldsymbol{A}_{i|j} \boldsymbol{x}_i^*.
\end{aligned} \tag{A.9}
$$

Combining (A.8) and (A.9) produces

$$
2 \sum_{i\in\mathcal{V}} (\nabla f_i^k(\boldsymbol{x}_i^{k+1}) - \nabla f_i(\boldsymbol{x}_i^\star))^T (\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^*)
$$

$$
\begin{aligned}
&+\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}_i}\boldsymbol{\lambda}_{i|j}^{\star,T}\boldsymbol{A}_{i|j}(\boldsymbol{x}_i^{k+1}-\boldsymbol{x}_i^*)\\
&+\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}_i}\boldsymbol{\lambda}_{j|i}^{k+1}\boldsymbol{A}_{i|j}(\boldsymbol{x}_i^{k+1}-\boldsymbol{x}_i^*)\\
&-\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}_i}\boldsymbol{A}_{j|i}\boldsymbol{x}_j^{k+1,T}(\boldsymbol{\lambda}_{i|j}^{k+1}-\boldsymbol{\lambda}_{i|j}^{\star})\\
&+\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}_i}(\boldsymbol{\lambda}_{i|j}^{k+1}-\boldsymbol{\lambda}_{i|j}^*)^T\boldsymbol{A}_{i|j}\boldsymbol{x}_i^*\\
=&\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}_i}\Big[\rho(\boldsymbol{A}_{j|i}\boldsymbol{x}_j^k-\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1})+\boldsymbol{\lambda}_{j|i}^{k+1}\\
&-\boldsymbol{\lambda}_{j|i}^k\Big]^T\boldsymbol{A}_{i|j}(\boldsymbol{x}_i^{k+1}-\boldsymbol{x}_i^*)+\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}_i}\Big[\boldsymbol{A}_{j|i}(\boldsymbol{x}_j^k-\boldsymbol{x}_j^{k+1})\\
&-\frac{1}{\rho}(\boldsymbol{\lambda}_{j|i}^k+\boldsymbol{\lambda}_{i|j}^{k+1})\Big]^T(\boldsymbol{\lambda}_{i|j}^{k+1}-\boldsymbol{\lambda}_{i|j}^{\star}).
\end{aligned}
\tag{A.10}
$$

Next, we simplify the left-hand side (LHS) of (A.10) as

$$
\begin{aligned}
&2\sum_{i\in\mathcal{V}}(\nabla f_i^k(\boldsymbol{x}_i^{k+1})-\nabla f_i(\boldsymbol{x}_i^{\star}))^T(\boldsymbol{x}_i^{k+1}-\boldsymbol{x}_i^*)\\
&\quad-2\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}_i}\boldsymbol{\lambda}_{i|j}^{\star,T}\boldsymbol{A}_{i|j}\boldsymbol{x}_i^*\\
=&2\sum_{i\in\mathcal{V}}(\nabla f_i(\boldsymbol{x}_i^k)+H_i^k(\boldsymbol{x}_i^{k+1}-\boldsymbol{x}^k)-\nabla f_i(\boldsymbol{x}_i^{\star}))^T(\boldsymbol{x}_i^{k+1}\\
&\quad-\boldsymbol{x}_i^*)-2\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}_i}\boldsymbol{\lambda}_{i|j}^{\star,T}\boldsymbol{A}_{i|j}\boldsymbol{x}_i^*\\
\overset{(a)}{=}&2\sum_{i\in\mathcal{V}}(\nabla f_i(\boldsymbol{x}_i^k)-\nabla f_i(\boldsymbol{x}_i^{k+1})+\nabla f_i(\boldsymbol{x}_i^{k+1})-\nabla f_i(\boldsymbol{x}_i^{\star}))^T\\
&\quad\cdot(\boldsymbol{x}_i^{k+1}-\boldsymbol{x}_i^*)-2\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}_i}\boldsymbol{\lambda}_{i|j}^{\star,T}\boldsymbol{A}_{i|j}\boldsymbol{x}_i^*\\
&+\sum_{i\in\mathcal{V}}\Big(\|\boldsymbol{x}_i^{k+1}-\boldsymbol{x}_i^k\|_{H_i^k}^2+\|\boldsymbol{x}_i^{k+1}-\boldsymbol{x}_i^*\|_{H_i^k}^2-\|\boldsymbol{x}_i^k-\boldsymbol{x}_i^*\|_{H_i^k}^2\Big)\\
\overset{(b)}{\geq}&+2\sum_{i\in\mathcal{V}}(\nabla f_i(\boldsymbol{x}_i^k)-\nabla f_i(\boldsymbol{x}_i^{k+1}))^T(\boldsymbol{x}_i^{k+1}-\boldsymbol{x}_i^*)\\
&\quad2\sum_{i\in\mathcal{V}}q_i\|\boldsymbol{x}_i^{k+1}-\boldsymbol{x}_i^*\|^2-2\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}_i}\boldsymbol{\lambda}_{i|j}^{\star,T}\boldsymbol{A}_{i|j}\boldsymbol{x}_i^*
\end{aligned}
$$

$$
\begin{aligned}
&+\sum_{i\in\mathcal{V}}\left(\|\boldsymbol{x}_i^{k+1}-\boldsymbol{x}_i^k\|_{H_i^k}^2+\|\boldsymbol{x}_i^{k+1}-\boldsymbol{x}_i^*\|_{H_i^k}^2-\|\boldsymbol{x}_i^k-\boldsymbol{x}_i^*\|_{H_i^k}^2\right)\\
&\geq-\sum_{i\in\mathcal{V}}\left(\frac{1}{c_i}\|(\nabla f_i(\boldsymbol{x}_i^k)-\nabla f_i(\boldsymbol{x}_i^{k+1}))\|^2+c_i\|\boldsymbol{x}_i^{k+1}-\boldsymbol{x}_i^*\|^2\right)\\
&\quad 2\sum_{i\in\mathcal{V}}q_i\|\boldsymbol{x}_i^{k+1}-\boldsymbol{x}_i^*\|^2-2\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}_i}\boldsymbol{\lambda}_{i|j}^{\star,T}\boldsymbol{A}_{i|j}\boldsymbol{x}_i^*\\
&+\sum_{i\in\mathcal{V}}\left(\|\boldsymbol{x}_i^{k+1}-\boldsymbol{x}_i^k\|_{H_i^k}^2+\|\boldsymbol{x}_i^{k+1}-\boldsymbol{x}_i^*\|_{H_i^k}^2-\|\boldsymbol{x}_i^k-\boldsymbol{x}_i^*\|_{H_i^k}^2\right)\\
&\overset{(c)}{\geq}\sum_{i\in\mathcal{V}}(2q_i-\boldsymbol{c}_i)\|\boldsymbol{x}_i^{k+1}-\boldsymbol{x}_i^*\|^2\\
&\quad+\sum_{i\in\mathcal{V}}(\boldsymbol{x}_i^{k+1}-\boldsymbol{x}_i^k)^T\left[H_i^k-\frac{Q_i}{c_i}\boldsymbol{I}\right](\boldsymbol{x}_i^{k+1}-\boldsymbol{x}_i^k)\\
&\quad+\sum_{i\in\mathcal{V}}\left(\|\boldsymbol{x}_i^{k+1}-\boldsymbol{x}_i^*\|_{H_i^k}^2-\|\boldsymbol{x}_i^k-\boldsymbol{x}_i^*\|_{H_i^k}^2\right),
\end{aligned}
\tag{A.11}
$$

where $c_i>0$ is a scalar parameter, step (a) uses

$$2(\boldsymbol{a}-\boldsymbol{b})^T\boldsymbol{P}(\boldsymbol{a}-\boldsymbol{c})=\|\boldsymbol{a}-\boldsymbol{b}\|_{\boldsymbol{P}}^2+\|\boldsymbol{a}-\boldsymbol{c}\|_{\boldsymbol{P}}^2-\|\boldsymbol{b}-\boldsymbol{c}\|_{\boldsymbol{P}}^2,$$

step (b) uses (A.1), and step (c) uses (A.2) and (A.4)-(A.5). The right-hand side (RHS) of (A.10) can be simplified as

$$
\begin{aligned}
&\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}_i}\left[\rho(\boldsymbol{A}_{j|i}\boldsymbol{x}_j^k-\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1})+\boldsymbol{\lambda}_{j|i}^{k+1}-\boldsymbol{\lambda}_{j|i}^k\right]^T\boldsymbol{A}_{i|j}(\boldsymbol{x}_i^{k+1}\\
&\quad-\boldsymbol{x}_i^*)+\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}_i}\left[\boldsymbol{A}_{j|i}(\boldsymbol{x}_j^k-\boldsymbol{x}_j^{k+1})-\frac{1}{\rho}(\boldsymbol{\lambda}_{j|i}^k+\boldsymbol{\lambda}_{i|j}^{k+1})\right]^T\\
&\quad\cdot(\boldsymbol{\lambda}_{i|j}^{k+1}-\boldsymbol{\lambda}_{i|j}^\star)\\
=&\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}_i}\left[\rho\boldsymbol{A}_{j|i}(\boldsymbol{x}_j^k-\boldsymbol{x}_j^{k+1})+\boldsymbol{\lambda}_{j|i}^{k+1}-\boldsymbol{\lambda}_{j|i}^k\right]^T\boldsymbol{A}_{i|j}(\boldsymbol{x}_i^{k+1}\\
&\quad-\boldsymbol{x}_i^*)+\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}_i}\left[\boldsymbol{A}_{j|i}(\boldsymbol{x}_j^k-\boldsymbol{x}_j^{k+1})+\frac{1}{\rho}(\boldsymbol{\lambda}_{j|i}^{k+1}-\boldsymbol{\lambda}_{j|i}^k)\right]^T\\
&\quad\cdot(\boldsymbol{\lambda}_{i|j}^{k+1}-\boldsymbol{\lambda}_{i|j}^\star)-\sum_{(i,j)\in E}\rho\|\boldsymbol{A}_{j|i}\boldsymbol{x}_j^{k+1}-\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1}\|^2
\end{aligned}
$$

$$
\begin{aligned}
&- \sum_{(i,j)\in E} 1/\rho \|\boldsymbol{\lambda}_{i|j}^{k+1} + \boldsymbol{\lambda}_{j|i}^{k+1}\|^2 \\
=&\sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{V}_i} \Big[\rho^{1/2}\boldsymbol{A}_{j|i}(\boldsymbol{x}_j^k - \boldsymbol{x}_j^{k+1}) + \rho^{-1/2}(\boldsymbol{\lambda}_{j|i}^{k+1} - \boldsymbol{\lambda}_{j|i}^{k})\Big]^T \\
&\qquad \cdot \Big[\rho^{1/2}\boldsymbol{A}_{i|j}(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^{*}) + \rho^{-1/2}(\boldsymbol{\lambda}_{i|j}^{k+1} - \boldsymbol{\lambda}_{i|j}^{\star})\Big] \\
&- \sum_{(i,j)\in E} \Big[\rho\|\boldsymbol{A}_{j|i}\boldsymbol{x}_j^{k+1} - \boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1}\|^2 - 1/\rho\|\boldsymbol{\lambda}_{i|j}^{k+1} + \boldsymbol{\lambda}_{j|i}^{k+1}\|^2\Big] \\
\overset{(a)}{=}&\frac{1}{2}\sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{V}_i} \Big[\|\rho^{-1/2}(\boldsymbol{\lambda}_{i|j}^{*} + \boldsymbol{\lambda}_{j|i}^{k}) + \rho^{1/2}(\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{*} - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^{k})\|^2 \\
&- \|\rho^{-1/2}(\boldsymbol{\lambda}_{i|j}^{*} + \boldsymbol{\lambda}_{j|i}^{k+1}) + \rho^{1/2}(\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{*} - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^{k+1})\|^2 \\
&+ \|\rho^{-1/2}(\boldsymbol{\lambda}_{i|j}^{k+1} + \boldsymbol{\lambda}_{j|i}^{k+1}) + \rho^{1/2}(\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1} - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^{k+1})\|^2 \\
&- \|\rho^{-1/2}(\boldsymbol{\lambda}_{i|j}^{k+1} + \boldsymbol{\lambda}_{j|i}^{k}) + \rho^{1/2}(\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1} - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^{k})\|^2\Big] \\
&- \sum_{(i,j)\in E} \Big[\rho\|\boldsymbol{A}_{j|i}\boldsymbol{x}_j^{k+1} - \boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1}\|^2 - 1/\rho\|\boldsymbol{\lambda}_{i|j}^{k+1} + \boldsymbol{\lambda}_{j|i}^{k+1}\|^2\Big] \\
=&\frac{1}{2}\sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{V}_i} \Big[\|\rho^{-1/2}(\boldsymbol{\lambda}_{i|j}^{*} + \boldsymbol{\lambda}_{j|i}^{k}) + \rho^{1/2}(\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{*} - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^{k})\|^2 \\
&- \|\rho^{-1/2}(\boldsymbol{\lambda}_{i|j}^{*} + \boldsymbol{\lambda}_{j|i}^{k+1}) + \rho^{1/2}(\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{*} - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^{k+1})\|^2 \\
&- \|\rho^{-1/2}(\boldsymbol{\lambda}_{i|j}^{k+1} + \boldsymbol{\lambda}_{j|i}^{k}) + \rho^{1/2}(\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1} - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^{k})\|^2\Big], \qquad \text{(A.12)}
\end{aligned}
$$

where step (a) uses

$$
\begin{aligned}
&(\boldsymbol{a}-\boldsymbol{b})^T(\boldsymbol{c}-\boldsymbol{d}) \\
&= \frac{1}{2}(\|\boldsymbol{a}+\boldsymbol{c}\|^2 - \|\boldsymbol{a}+\boldsymbol{d}\|^2 - \|\boldsymbol{b}+\boldsymbol{c}\|^2 + \|\boldsymbol{b}+\boldsymbol{d}\|^2).
\end{aligned}
$$

Combining (A.10)-(A.12) yields equation (5.13).

A.2 QA-PDMM Convergence Inequality 2

Invoking Lemma 6 with $\boldsymbol{x}_i = \boldsymbol{x}_i^{\star}$ and rearranging the expression, we obtain

$$
\frac{h_i}{2}\|\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^{\star}\|^2 + f_i(\boldsymbol{x}_i^{k+1}) - f_i(\boldsymbol{x}_i^{\star})
$$

$$\leq (\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^\star)^T \sum_{j\in\mathcal{N}_i} \boldsymbol{A}_{i|j}^T \boldsymbol{\lambda}_{i|j}^{k+1} + \frac{h_i}{2}\|\boldsymbol{x}_i^k - \boldsymbol{x}_i^\star\|^2 \tag{A.13}$$

Summing the above inequality over $i \in \mathcal{V}$ and plugging the expression for $\boldsymbol{\lambda}_{i|j}^{k+1}$ produces

$$\begin{aligned}
&\sum_{i\in\mathcal{V}} \left[\frac{h_i}{2}\|\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^\star\|^2 + f_i(\boldsymbol{x}_i^{k+1}) - f_i(\boldsymbol{x}_i^\star)\right] \\
&\leq \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i} (\boldsymbol{P}_{ij}(\boldsymbol{A}_{j|i}\boldsymbol{x}_j^k - \boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1}) - \boldsymbol{\lambda}_{j|i}^k)^T \\
&\qquad\qquad \cdot \boldsymbol{A}_{i|j}(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^\star) + \sum_{i\in\mathcal{V}} \frac{h_i}{2}\|\boldsymbol{x}_i^k - \boldsymbol{x}_i^\star\|^2 \\
&= \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i} \left(\boldsymbol{P}_{ij}(\boldsymbol{A}_{j|i}\boldsymbol{x}_j^k - \boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1}) + \boldsymbol{\lambda}_{j|i}^{k+1} - \boldsymbol{\lambda}_{j|i}^k\right)^T \\
&\qquad\qquad \cdot \boldsymbol{A}_{i|j}(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^\star) + \sum_{i\in\mathcal{V}} \frac{h_i}{2}\|\boldsymbol{x}_i^k - \boldsymbol{x}_i^\star\|^2 \\
&\quad - \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i} \boldsymbol{\lambda}_{j|i}^{k+1}\boldsymbol{A}_{i|j}(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^\star).
\end{aligned} \tag{A.14}$$

Next we derive a different upper bound for the quantity on the left hand side of (A.14). To do so, we note from (5.2) that $\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1}$ can be represented in terms of $\boldsymbol{\lambda}_{i|j}^{k+1}$ as

$$\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1} = A_{j|i}\boldsymbol{x}_j^k - \boldsymbol{P}_{ij}^{-1}(\boldsymbol{\lambda}_{j|i}^k + \boldsymbol{\lambda}_{i|j}^{k+1}) \;\; i \in \mathcal{V}, j \in \mathcal{N}_i, \tag{A.15}$$

Similarly to the derivation of (A.14), we sum (A.19) over all $i \in \mathcal{V}$ and plugging the expression (A.15) for $\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1}$ where appropriate, which are given by

$$\begin{aligned}
&\sum_{i\in\mathcal{V}} \left[\frac{h_i}{2}\|\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^\star\|^2 + f_i(\boldsymbol{x}_i^{k+1}) - f_i(\boldsymbol{x}_i^\star)\right] \\
&\leq \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i} \boldsymbol{\lambda}_{i|j}^{k+1,T}\boldsymbol{A}_{i|j}\left(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^\star\right) + \sum_{i\in\mathcal{V}} \frac{h_i}{2}\|\boldsymbol{x}_i^k - \boldsymbol{x}_i^\star\|^2 \\
&= \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i} (\boldsymbol{\lambda}_{i|j}^{k+1} - \boldsymbol{\lambda}_{i|j}^\star)^T\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1} - \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i} \boldsymbol{\lambda}_{i|j}^{k+1,T}\boldsymbol{A}_{i|j}\boldsymbol{x}_i^\star
\end{aligned}$$

$$
\begin{aligned}
&+\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\boldsymbol{\lambda}_{i|j}^{\star,T}\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1}+\sum_{i\in\mathcal{V}}\frac{h_i}{2}\|\boldsymbol{x}_i^k-\boldsymbol{x}_i^\star\|^2\\
=&\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\left[A_{j|i}\boldsymbol{x}_j^k-\boldsymbol{P}_{ij}^{-1}(\boldsymbol{\lambda}_{j|i}^k+\boldsymbol{\lambda}_{i|j}^{k+1})\right]^T(\boldsymbol{\lambda}_{i|j}^{k+1}-\boldsymbol{\lambda}_{i|j}^\star)\\
&-\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\boldsymbol{\lambda}_{i|j}^{k+1,T}\boldsymbol{A}_{i|j}\boldsymbol{x}_i^\star+\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\boldsymbol{\lambda}_{i|j}^{\star,T}\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1}\\
&+\sum_{i\in\mathcal{V}}\frac{h_i}{2}\|\boldsymbol{x}_i^k-\boldsymbol{x}_i^\star\|^2\\
=&\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\left[A_{j|i}(\boldsymbol{x}_j^k-\boldsymbol{x}_j^{k+1})-\boldsymbol{P}_{ij}^{-1}(\boldsymbol{\lambda}_{j|i}^k+\boldsymbol{\lambda}_{i|j}^{k+1})\right]^T\\
&\qquad\cdot(\boldsymbol{\lambda}_{i|j}^{k+1}-\boldsymbol{\lambda}_{i|j}^\star)+\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}(\boldsymbol{A}_{j|i}\boldsymbol{x}_j^{k+1})^T(\boldsymbol{\lambda}_{i|j}^{k+1}-\boldsymbol{\lambda}_{i|j}^\star)\\
&-\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\boldsymbol{\lambda}_{i|j}^{k+1,T}\boldsymbol{A}_{i|j}\boldsymbol{x}_i^\star+\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\boldsymbol{\lambda}_{i|j}^{\star,T}\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1}\\
&+\sum_{i\in\mathcal{V}}\frac{h_i}{2}\|\boldsymbol{x}_i^k-\boldsymbol{x}_i^\star\|^2
\end{aligned}
\tag{A.16}
$$

Combining (A.14)-(A.22) produces

$$
\begin{aligned}
&2\sum_{i\in\mathcal{V}}\Big[\frac{h_i}{2}\|\boldsymbol{x}_i^{k+1}-\boldsymbol{x}_i^\star\|^2+f_i(\boldsymbol{x}_i^{k+1})-f_i(\boldsymbol{x}_i^\star)\\
&\qquad-\boldsymbol{x}_i^{k+1,T}\sum_{j\in\mathcal{N}_i}A_{i|j}^T\boldsymbol{\lambda}_{i|j}^\star\Big]\\
\le&\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\Big(\boldsymbol{P}_{ij}(\boldsymbol{A}_{j|i}\boldsymbol{x}_j^k-\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1})+\boldsymbol{\lambda}_{j|i}^{k+1}-\boldsymbol{\lambda}_{j|i}^k\Big)^T\\
&\qquad\cdot\boldsymbol{A}_{i|j}(\boldsymbol{x}_i^{k+1}-\boldsymbol{x}_i^\star)+\sum_{i\in\mathcal{V}}\frac{h_i}{2}\|\boldsymbol{x}_i^k-\boldsymbol{x}_i^\star\|^2\\
&+\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\left[A_{j|i}(\boldsymbol{x}_j^k-\boldsymbol{x}_j^{k+1})-\boldsymbol{P}_{ij}^{-1}(\boldsymbol{\lambda}_{j|i}^k+\boldsymbol{\lambda}_{i|j}^{k+1})\right]^T\\
&\qquad\cdot(\boldsymbol{\lambda}_{i|j}^{k+1}-\boldsymbol{\lambda}_{i|j}^\star)+\sum_{i\in\mathcal{V}}\frac{h_i}{2}\|\boldsymbol{x}_i^k-\boldsymbol{x}_i^\star\|^2\\
=&\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\Big(\boldsymbol{P}_{ij}\boldsymbol{A}_{j|i}(\boldsymbol{x}_j^k-\boldsymbol{x}_j^{k+1})+\boldsymbol{\lambda}_{j|i}^{k+1}-\boldsymbol{\lambda}_{j|i}^k\Big)^T
\end{aligned}
$$

$$
\begin{aligned}
&\cdot \boldsymbol{A}_{i|j}(\boldsymbol{x}_i^{k+1}-\boldsymbol{x}_i^{\star}) - \sum_{(i,j)\in\mathcal{E}} \|A_{j|i}\boldsymbol{x}_j^{k+1} - A_{i|j}\boldsymbol{x}_i^{k+1}\|^2_{\boldsymbol{P}_{ij}} \\
&+ \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\Big[A_{j|i}(\boldsymbol{x}_j^{k}-\boldsymbol{x}_j^{k+1}) - \boldsymbol{P}_{ij}^{-1}(\boldsymbol{\lambda}_{j|i}^{k}-\boldsymbol{\lambda}_{j|i}^{k+1})\Big]^T \\
&\cdot (\boldsymbol{\lambda}_{i|j}^{k+1}-\boldsymbol{\lambda}_{i|j}^{\star}) - \sum_{(i,j)\in\mathcal{E}} \|\boldsymbol{\lambda}_{i|j}^{k+1}+\boldsymbol{\lambda}_{j|i}^{k+1}\|^2_{\boldsymbol{P}_{ij}^{-1}} \\
&+ 2\sum_{i\in\mathcal{V}}\frac{h_i}{2}\|\boldsymbol{x}_i^{k}-\boldsymbol{x}_i^{\star}\|^2 \\
=& \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\Big(\boldsymbol{P}_{ij}^{\frac{1}{2}}\boldsymbol{A}_{j|i}(\boldsymbol{x}_j^{k}-\boldsymbol{x}_j^{k+1}) + \boldsymbol{P}_{ij}^{-\frac{1}{2}}(\boldsymbol{\lambda}_{j|i}^{k+1}-\boldsymbol{\lambda}_{j|i}^{k})\Big)^T \\
&\quad\cdot\Big(\boldsymbol{P}_{ij}^{\frac{1}{2}}\boldsymbol{A}_{i|j}(\boldsymbol{x}_i^{k+1}-\boldsymbol{x}_i^{\star}) + \boldsymbol{P}_{ij}^{-\frac{1}{2}}(\boldsymbol{\lambda}_{i|j}^{k+1}-\boldsymbol{\lambda}_{i|j}^{\star})\Big) \\
&- \sum_{(i,j)\in\mathcal{E}} \|A_{j|i}\boldsymbol{x}_j^{k+1} - A_{i|j}\boldsymbol{x}_i^{k+1}\|^2_{\boldsymbol{P}_{ij}} - \sum_{(i,j)\in\mathcal{E}} \|\boldsymbol{\lambda}_{i|j}^{k+1}+\boldsymbol{\lambda}_{j|i}^{k+1}\|^2_{\boldsymbol{P}_{ij}^{-1}} \\
&+ 2\sum_{i\in\mathcal{V}}\frac{h_i}{2}\|\boldsymbol{x}_i^{k}-\boldsymbol{x}_i^{\star}\|^2 \\
\overset{(a)}{=}& \frac{1}{2}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\Big[\|\boldsymbol{P}_{ij}^{-1/2}(\boldsymbol{\lambda}_{i|j}^{k+1}+\boldsymbol{\lambda}_{j|i}^{k+1}) + \boldsymbol{P}_{ij}^{1/2}(\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1} - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^{k+1})\|^2 \\
&\quad - \|\boldsymbol{P}_{ij}^{-1/2}(\boldsymbol{\lambda}_{i|j}^{*}+\boldsymbol{\lambda}_{j|i}^{k+1}) + \boldsymbol{P}_{ij}^{1/2}(\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{*} - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^{k+1})\|^2 \\
&\quad + \|\boldsymbol{P}_{ij}^{-1/2}(\boldsymbol{\lambda}_{i|j}^{*}+\boldsymbol{\lambda}_{j|i}^{k}) + \boldsymbol{P}_{ij}^{1/2}(\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{*} - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^{k})\|^2 \\
&\quad - \|\boldsymbol{P}_{ij}^{-1/2}(\boldsymbol{\lambda}_{i|j}^{k+1}+\boldsymbol{\lambda}_{j|i}^{k}) + \boldsymbol{P}_{ij}^{1/2}(\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1} - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^{k})\|^2\Big] \\
&- \sum_{(i,j)\in\mathcal{E}} \|A_{j|i}\boldsymbol{x}_j^{k+1} - A_{i|j}\boldsymbol{x}_i^{k+1}\|^2_{\boldsymbol{P}_{ij}} - \sum_{(i,j)\in\mathcal{E}} \|\boldsymbol{\lambda}_{i|j}^{k+1}+\boldsymbol{\lambda}_{j|i}^{k+1}\|^2_{\boldsymbol{P}_{ij}^{-1}} \\
&+ 2\sum_{i\in\mathcal{V}}\frac{h_i}{2}\|\boldsymbol{x}_i^{k}-\boldsymbol{x}_i^{\star}\|^2 \\
\overset{(b)}{=}& \frac{1}{2}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\Big[- \|\boldsymbol{P}_{ij}^{-1/2}(\boldsymbol{\lambda}_{i|j}^{*}+\boldsymbol{\lambda}_{j|i}^{k+1}) + \boldsymbol{P}_{ij}^{1/2}(\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{*} - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^{k+1})\|^2 \\
&\quad + \|\boldsymbol{P}_{ij}^{-1/2}(\boldsymbol{\lambda}_{i|j}^{*}+\boldsymbol{\lambda}_{j|i}^{k}) + \boldsymbol{P}_{ij}^{1/2}(\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{*} - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^{k})\|^2 \\
&\quad - \|\boldsymbol{P}_{ij}^{-1/2}(\boldsymbol{\lambda}_{i|j}^{k+1}+\boldsymbol{\lambda}_{j|i}^{k}) + \boldsymbol{P}_{ij}^{1/2}(\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1} - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^{k})\|^2\Big] \\
&+ 2\sum_{i\in\mathcal{V}}\frac{h_i}{2}\|\boldsymbol{x}_i^{k}-\boldsymbol{x}_i^{\star}\|^2
\end{aligned}
\tag{A.17}
$$

where step (a) follows from the the identity

$$\begin{aligned}&(\boldsymbol{a}-\boldsymbol{b})^T(\boldsymbol{c}-\boldsymbol{d})\\&=\frac{1}{2}(\|\boldsymbol{a}+\boldsymbol{c}\|^2-\|\boldsymbol{a}+\boldsymbol{d}\|^2-\|\boldsymbol{b}+\boldsymbol{c}\|^2+\|\boldsymbol{b}+\boldsymbol{d}\|^2).\end{aligned}$$

and step (b) uses (5.6) and (5.8)-(5.9), similar to the proof for Lemma 8 in [157].

Next we prove the lower bound of (5.22).

$$\begin{aligned}&2\sum_{i\in\mathcal{V}}\Big[f_i(\boldsymbol{x}_i^{k+1})-f_i(\boldsymbol{x}_i^\star)-\boldsymbol{x}_i^{k+1,T}\sum_{j\in\mathcal{N}_i}A_{i|j}^T\boldsymbol{\lambda}_{i|j}^\star\Big]\\&\overset{(a)}{\geq}2\sum_{i\in\mathcal{V}}\Big[-f_i^*\bigg(\sum_{i\in\mathcal{N}_i}\boldsymbol{A}_{i|j}^T\boldsymbol{\lambda}_{i|j}^\star\bigg)-f_i(\boldsymbol{x}_i^\star)\Big]\\&\overset{(b)}{=}0,\end{aligned}$$

where $f_i^*(\cdot)$ is the conjugate function of $f_i(\cdot)$, step (a) uses Fenchel's inequality (see [16]), and step (b) uses (5.8)-(5.9) and the definition of the conjugate functions $\{f_i^*|i\in\mathcal{V}\}$. The equality in step (a) holds when (5.23) is satisfied.

A.3 FSQA-PDMM Convergence Inequality

This proof of this lemma is similar to that in Appendix A.2 for QA-PDMM. Invoking Lemma 5 and 6 with $\boldsymbol{x}_i=\boldsymbol{x}_i^\star$ and rearranging the expression, we obtain

$$\begin{aligned}&(\boldsymbol{x}_i^{k+1}-\boldsymbol{x}_i^\star)^T\sum_{j\in\mathcal{N}_i}\boldsymbol{A}_{i|j}^T\boldsymbol{\lambda}_{i|j}^{k+1}+\frac{h_i}{2}\|\boldsymbol{x}_i^k-\boldsymbol{x}_i^\star\|^2\\&\geq\frac{h_i}{2}\|\boldsymbol{x}_i^{k+1}-\boldsymbol{x}_i^\star\|^2+f_i(\boldsymbol{x}_i^{k+1})-f_i(\boldsymbol{x}_i^\star)\ \ i\in\mathcal{U}\end{aligned}\tag{A.18}$$

and

$$(\boldsymbol{x}_i^{k+1}-\boldsymbol{x}_i^\star)^T\sum_{j\in\mathcal{N}_i}\boldsymbol{A}_{i|j}^T\boldsymbol{\lambda}_{i|j}^{k+1}\geq f_i(\boldsymbol{x}_i^{k+1})-f_i(\boldsymbol{x}_i^\star)\ \ i\in\mathcal{W}\tag{A.19}$$

Summing the above two inequalities over $i \in \mathcal{U} \cup \mathcal{W}$ and plugging the expression for $\boldsymbol{\lambda}_{i|j}^{k+1}$ produces

$$\begin{aligned}
&\sum_{i\in\mathcal{V}} \big[f_i(\boldsymbol{x}_i^{k+1}) - f_i(\boldsymbol{x}_i^\star)\big] + \sum_{i\in\mathcal{U}} \frac{h_i}{2}\|\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^\star\|^2 \\
&\leq \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{N}_i} (\boldsymbol{P}_{ij}(\boldsymbol{A}_{j|i}\boldsymbol{x}_j^k - \boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1}) - \boldsymbol{\lambda}_{j|i}^k)^T \\
&\qquad\qquad \cdot \boldsymbol{A}_{i|j}(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^\star) + \sum_{i\in\mathcal{U}} \frac{h_i}{2}\|\boldsymbol{x}_i^k - \boldsymbol{x}_i^\star\|^2 \\
&= \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{N}_i} \Big(\boldsymbol{P}_{ij}(\boldsymbol{A}_{j|i}\boldsymbol{x}_j^k - \boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1}) + \boldsymbol{\lambda}_{j|i}^{k+1} - \boldsymbol{\lambda}_{j|i}^k\Big)^T \\
&\qquad\qquad \cdot \boldsymbol{A}_{i|j}(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^\star) + \sum_{i\in\mathcal{U}} \frac{h_i}{2}\|\boldsymbol{x}_i^k - \boldsymbol{x}_i^\star\|^2 \\
&\quad - \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{N}_i} \boldsymbol{\lambda}_{j|i}^{k+1} \boldsymbol{A}_{i|j}(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^\star).
\end{aligned} \tag{A.20}$$

Next we derive a different upper bound for the quantity on the left hand side of (A.20). To do so, we note from (5.2) that $\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1}$ can be represented in terms of $\boldsymbol{\lambda}_{i|j}^{k+1}$ as

$$\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1} = A_{j|i}\boldsymbol{x}_j^k - \boldsymbol{P}_{ij}^{-1}(\boldsymbol{\lambda}_{j|i}^k + \boldsymbol{\lambda}_{i|j}^{k+1}) \;\; i \in \mathcal{V}, j \in \mathcal{N}_i, \tag{A.21}$$

Similarly to the derivation of (A.20), we sum (A.19) over all $i \in \mathcal{U} \cup \mathcal{W}$ and plugging the expression (A.21) for $\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1}$ where appropriate, which are given by

$$\begin{aligned}
&\sum_{i\in\mathcal{V}} \big[f_i(\boldsymbol{x}_i^{k+1}) - f_i(\boldsymbol{x}_i^\star)\big] + \sum_{i\in\mathcal{U}} \frac{h_i}{2}\|\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^\star\|^2 \\
&\leq \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{N}_i} \boldsymbol{\lambda}_{i|j}^{k+1,T} \boldsymbol{A}_{i|j}\left(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^\star\right) + \sum_{i\in\mathcal{U}} \frac{h_i}{2}\|\boldsymbol{x}_i^k - \boldsymbol{x}_i^\star\|^2 \\
&= \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{N}_i} (\boldsymbol{\lambda}_{i|j}^{k+1} - \boldsymbol{\lambda}_{i|j}^\star)^T \boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1} - \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{N}_i} \boldsymbol{\lambda}_{i|j}^{k+1,T} \boldsymbol{A}_{i|j}\boldsymbol{x}_i^\star \\
&\quad + \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{N}_i} \boldsymbol{\lambda}_{i|j}^{\star,T} \boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1} + \sum_{i\in\mathcal{U}} \frac{h_i}{2}\|\boldsymbol{x}_i^k - \boldsymbol{x}_i^\star\|^2
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\left[A_{j|i}\boldsymbol{x}_j^k - \boldsymbol{P}_{ij}^{-1}(\boldsymbol{\lambda}_{j|i}^k + \boldsymbol{\lambda}_{i|j}^{k+1})\right]^T(\boldsymbol{\lambda}_{i|j}^{k+1} - \boldsymbol{\lambda}_{i|j}^\star)\\
&\quad -\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\boldsymbol{\lambda}_{i|j}^{k+1,T}\boldsymbol{A}_{i|j}\boldsymbol{x}_i^\star + \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\boldsymbol{\lambda}_{i|j}^{\star,T}\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1}\\
&\quad +\sum_{i\in\mathcal{U}}\frac{h_i}{2}\|\boldsymbol{x}_i^k - \boldsymbol{x}_i^\star\|^2\\
&= \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\left[A_{j|i}(\boldsymbol{x}_j^k - \boldsymbol{x}_j^{k+1}) - \boldsymbol{P}_{ij}^{-1}(\boldsymbol{\lambda}_{j|i}^k + \boldsymbol{\lambda}_{i|j}^{k+1})\right]^T\\
&\qquad \cdot(\boldsymbol{\lambda}_{i|j}^{k+1} - \boldsymbol{\lambda}_{i|j}^\star) + \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}(\boldsymbol{A}_{j|i}\boldsymbol{x}_j^{k+1})^T(\boldsymbol{\lambda}_{i|j}^{k+1} - \boldsymbol{\lambda}_{i|j}^\star)\\
&\quad -\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\boldsymbol{\lambda}_{i|j}^{k+1,T}\boldsymbol{A}_{i|j}\boldsymbol{x}_i^\star + \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\boldsymbol{\lambda}_{i|j}^{\star,T}\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1}\\
&\quad +\sum_{i\in\mathcal{U}}\frac{h_i}{2}\|\boldsymbol{x}_i^k - \boldsymbol{x}_i^\star\|^2
\end{aligned}
\tag{A.22}
$$

Combining (A.20)-(A.22) produces

$$
\begin{aligned}
&2\sum_{i\in\mathcal{V}}\left[f_i(\boldsymbol{x}_i^{k+1}) - f_i(\boldsymbol{x}_i^\star) - \boldsymbol{x}_i^{k+1,T}\sum_{j\in\mathcal{N}_i}A_{i|j}^T\boldsymbol{\lambda}_{i|j}^\star\right]\\
&+\sum_{i\in\mathcal{U}}h_i\|\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^\star\|^2\\
&\le \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\left(\boldsymbol{P}_{ij}(\boldsymbol{A}_{j|i}\boldsymbol{x}_j^k - \boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1}) + \boldsymbol{\lambda}_{j|i}^{k+1} - \boldsymbol{\lambda}_{j|i}^k\right)^T\\
&\qquad\qquad \cdot\boldsymbol{A}_{i|j}(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^\star) + \sum_{i\in\mathcal{U}}h_i\|\boldsymbol{x}_i^k - \boldsymbol{x}_i^\star\|^2\\
&\quad +\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\left[A_{j|i}(\boldsymbol{x}_j^k - \boldsymbol{x}_j^{k+1}) - \boldsymbol{P}_{ij}^{-1}(\boldsymbol{\lambda}_{j|i}^k + \boldsymbol{\lambda}_{i|j}^{k+1})\right]^T\\
&\qquad\cdot(\boldsymbol{\lambda}_{i|j}^{k+1} - \boldsymbol{\lambda}_{i|j}^\star)\\
&= \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\left(\boldsymbol{P}_{ij}\boldsymbol{A}_{j|i}(\boldsymbol{x}_j^k - \boldsymbol{x}_j^{k+1}) + \boldsymbol{\lambda}_{j|i}^{k+1} - \boldsymbol{\lambda}_{j|i}^k\right)^T\\
&\quad\cdot\boldsymbol{A}_{i|j}(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^\star) - \sum_{(i,j)\in\mathcal{E}}\|A_{j|i}\boldsymbol{x}_j^{k+1} - A_{i|j}\boldsymbol{x}_i^{k+1}\|_{\boldsymbol{P}_{ij}}^2\\
&\quad +\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\left[A_{j|i}(\boldsymbol{x}_j^k - \boldsymbol{x}_j^{k+1}) - \boldsymbol{P}_{ij}^{-1}(\boldsymbol{\lambda}_{j|i}^k - \boldsymbol{\lambda}_{j|i}^{k+1})\right]^T
\end{aligned}
$$

$$
\begin{aligned}
&\cdot (\boldsymbol{\lambda}_{i|j}^{k+1} - \boldsymbol{\lambda}_{i|j}^{\star}) - \sum_{(i,j)\in\mathcal{E}} \|\boldsymbol{\lambda}_{i|j}^{k+1} + \boldsymbol{\lambda}_{j|i}^{k+1}\|^2_{\boldsymbol{P}_{ij}^{-1}} \\
&+ \sum_{i\in\mathcal{U}} h_i \|\boldsymbol{x}_i^k - \boldsymbol{x}_i^{\star}\|^2 \\
&= \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{N}_i} \Big(\boldsymbol{P}_{ij}^{\frac{1}{2}} \boldsymbol{A}_{j|i}(\boldsymbol{x}_j^k - \boldsymbol{x}_j^{k+1}) + \boldsymbol{P}_{ij}^{-\frac{1}{2}}(\boldsymbol{\lambda}_{j|i}^{k+1} - \boldsymbol{\lambda}_{j|i}^{k})\Big)^T \\
&\qquad \cdot \Big(\boldsymbol{P}_{ij}^{\frac{1}{2}} \boldsymbol{A}_{i|j}(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^{\star}) + \boldsymbol{P}_{ij}^{-\frac{1}{2}}(\boldsymbol{\lambda}_{i|j}^{k+1} - \boldsymbol{\lambda}_{i|j}^{\star})\Big) \\
&- \sum_{(i,j)\in\mathcal{E}} \|A_{j|i}\boldsymbol{x}_j^{k+1} - A_{i|j}\boldsymbol{x}_i^{k+1}\|^2_{\boldsymbol{P}_{ij}} - \sum_{(i,j)\in\mathcal{E}} \|\boldsymbol{\lambda}_{i|j}^{k+1} + \boldsymbol{\lambda}_{j|i}^{k+1}\|^2_{\boldsymbol{P}_{ij}^{-1}} \\
&+ \sum_{i\in\mathcal{V}} h_i \|\boldsymbol{x}_i^k - \boldsymbol{x}_i^{\star}\|^2 \\
&\overset{(a)}{=} \frac{1}{2} \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{N}_i} \Big[\|\boldsymbol{P}_{ij}^{-1/2}(\boldsymbol{\lambda}_{i|j}^{k+1} + \boldsymbol{\lambda}_{j|i}^{k+1}) \\
&\qquad\qquad + \boldsymbol{P}_{ij}^{1/2}(\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1} - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^{k+1})\|^2 \\
&\qquad - \|\boldsymbol{P}_{ij}^{-1/2}(\boldsymbol{\lambda}_{i|j}^{*} + \boldsymbol{\lambda}_{j|i}^{k+1}) + \boldsymbol{P}_{ij}^{1/2}(\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{*} - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^{k+1})\|^2 \\
&\qquad + \|\boldsymbol{P}_{ij}^{-1/2}(\boldsymbol{\lambda}_{i|j}^{*} + \boldsymbol{\lambda}_{j|i}^{k}) + \boldsymbol{P}_{ij}^{1/2}(\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{*} - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^{k})\|^2 \\
&\qquad - \|\boldsymbol{P}_{ij}^{-1/2}(\boldsymbol{\lambda}_{i|j}^{k+1} + \boldsymbol{\lambda}_{j|i}^{k}) + \boldsymbol{P}_{ij}^{1/2}(\boldsymbol{A}_{i|j}\boldsymbol{x}_i^{k+1} - \boldsymbol{A}_{j|i}\boldsymbol{x}_j^{k})\|^2\Big] \\
&- \sum_{(i,j)\in\mathcal{E}} \|A_{j|i}\boldsymbol{x}_j^{k+1} - A_{i|j}\boldsymbol{x}_i^{k+1}\|^2_{\boldsymbol{P}_{ij}} - \sum_{(i,j)\in\mathcal{E}} \|\boldsymbol{\lambda}_{i|j}^{k+1} + \boldsymbol{\lambda}_{j|i}^{k+1}\|^2_{\boldsymbol{P}_{ij}^{-1}} \\
&+ \sum_{i\in\mathcal{U}} h_i \|\boldsymbol{x}_i^k - \boldsymbol{x}_i^{\star}\|^2, \qquad \text{(A.23)}
\end{aligned}
$$

where step (a) follows from the the identity

$$
\begin{aligned}
&(\boldsymbol{a} - \boldsymbol{b})^T(\boldsymbol{c} - \boldsymbol{d}) \\
&= \frac{1}{2}(\|\boldsymbol{a} + \boldsymbol{c}\|^2 - \|\boldsymbol{a} + \boldsymbol{d}\|^2 - \|\boldsymbol{b} + \boldsymbol{c}\|^2 + \|\boldsymbol{b} + \boldsymbol{d}\|^2).
\end{aligned}
$$

Reformulating (A.23) produces (6.23).

Next we prove the lower bound of (6.23).

$$
2\sum_{i\in\mathcal{V}} \Big[f_i(\boldsymbol{x}_i^{k+1}) - f_i(\boldsymbol{x}_i^{\star}) - \boldsymbol{x}_i^{k+1,T} \sum_{j\in\mathcal{V}_i} A_{i|j}^T \boldsymbol{\lambda}_{i|j}^{\star}\Big]
$$

$$\begin{aligned}
&\overset{(a)}{\geq} 2\sum_{i\in\mathcal{V}}\Big[-f_i^*\left(\sum_{i\in\mathcal{V}_i}\boldsymbol{A}_{i|j}^T\boldsymbol{\lambda}_{i|j}^\star\right)-f_i(\boldsymbol{x}_i^\star)\Big]\\
&\overset{(b)}{=} 0,
\end{aligned}$$

where $f_i^*(\cdot)$ is the conjugate function of $f_i(\cdot)$, step (a) uses Fenchel's inequality (see [16]), and step (b) uses (6.15)-(6.16) and the definition of the conjugate functions $\{f_i^*|i\in\mathcal{V}\}$. The equality in step (a) holds when (5.23) is satisfied.

Bibliography

[1] AGARWALA, A., AGRAWALA, M., COHEN, M., SALESIN, D., AND SZELISKI, R. Photographing long scenes with multi-viewpoint panoramas. *ACM Trans. Graph. 25*, 3 (July 2006), 853–861.

[2] ALDOUS, D., AND STEELE, J. The objective method: Probabilistic combinatorial optimization and local weak convergence. In *Probability on Discrete Structures*, H. Kesten, Ed., vol. 110 of *Encyclopaedia of Mathematical Sciences*. Springer Berlin Heidelberg, 2004, pp. 1–72.

[3] ANTONINI, M., BARLAUD, M., MATHIEU, P., AND DAUBECHIES, I. Image coding using wavelet transform. *IEEE Transactions on Image Processing 1*, 2 (Apr 1992), 205–220.

[4] AU, J. K. *An ab initio approach to the inverse problem-based design of photonic bandgap devices*. PhD thesis, California Institute of Technology, 2007.

[5] BACH, F., JENATTON, R., MAIRAL, J., OBOZINSKI, G., ET AL. Convex optimization with sparsity-inducing norms. *Optimization for Machine Learning* (2011), 19–53.

[6] BECK, A., AND TEBOULLE, M. A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM J. Imaging Sciences 2*, 1 (2009), 183–202.

[7] BENESTY, J., C. J. H. Y. *Microphone Array Signal Processing*, 1 ed. Springer-Verlag Berlin Heidelberg, 2008.

[8] BERTRAND, A., AND MOONEN, M. Distributed adaptive estimation of node-specific signals in wireless sensor networks with a tree topology. *Signal Processing, IEEE Transactions on 59*, 5 (2011), 2196–2210.

[9] BERTRAND, A., AND MOONEN, M. Distributed node-specific LCMV beamforming in wireless sensor networks. *Signal Processing, IEEE Transactions on 60*, 1 (2012), 233–246.

[10] BERTRAND, A., AND MOONEN, M. Distributed LCMV beamforming in a wireless sensor network with single-channel per-node signal transmission. *Signal Processing, IEEE Transactions on 61*, 13 (2013), 3447–3459.

[11] BERTSEKAS, D. P. *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, 1982.

[12] BISHOP, C. M. *Pattern Recognition and Machine Learning*, 1 ed. Springer, 2006.

[13] BOYD, S., GHOSH, A., PRABHAKAR, B., AND SHAH, D. Analysis and optimization of randomized gossip algorithms. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on* (Dec 2004), vol. 5, pp. 5310–5315 Vol.5.

[14] BOYD, S., GHOSH, A., PRABHAKAR, B., AND SHAH, D. Randomized Gossip Algorithms. *IEEE Trans. Information Theory 52*, 6 (2006), 2508–2530.

[15] BOYD, S., PARIKH, N., CHU, E., PELEATO, B., AND ECKSTEIN, J. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *In Foundations and Trends in Machine Learning 3*, 1 (2011), 1–122.

[16] BOYD, S., AND VANDENBERGHE, L. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[17] BRANDSTEIN, M., AND WARD, D. E. *Microphone Arrays: Signal Processing Techniques and Applications*, 1 ed. Springer-Verlag Berlin Heidelberg, 2001.

[18] BRANDWOOD, D. A complex gradient operator and its application in adaptive array theory. *Microwaves, Optics and Antennas, IEE Proceedings H 130*, 1 (February 1983), 11–16.

[19] CAMPBELL, M. E., AND AHMED, N. R. Distributed data fusion: Neighbors, rumors, and the art of collective knowledge. *IEEE Control Systems 36*, 4 (Aug 2016), 83–109.

[20] CAPON, J. High-resolution frequency-wavenumber spectrum analysis. *Proceedings of the IEEE 57*, 8 (Aug 1969), 1408–1418.

[21] CEVHER, V., BECKER, S., AND SCHMIDT, M. Convex optimization for big data: Scalable, randomized, and parallel algorithms for big data analytics. *Signal Processing Magazine, IEEE 31*, 5 (2014), 32–43.

[22] CHANG, N. B., BAI, K., IMEN, S., CHEN, C. F., AND GAO, W. Multisensor satellite image fusion and networking for all-weather environmental monitoring. *IEEE Systems Journal PP*, 99 (2016), 1–17.

[23] CHANG, T.-H., HONG, M., AND WANG, X. Multi-agent distributed optimization via inexact consensus ADMM. *IEEE Transactions on Signal Processing 63*, 2 (2015), 482–497.

[24] CHARALAMBOUS, T., YUAN, Y., YANG, T., PAN, W., HADJICOSTIS, C. N., AND JOHANSSON, M. Distributed Finite-Time Average Consensus in Digraphs in the Presence of Time Delays. *IEEE Trans. Control of Network Systems 2*, 4 (2015), 370–381.

[25] CHEN, G., AND TEBOULLE, M. A proximal-based decomposition method for convex minimization problems. *Mathematical Programming 64* (1994), 81–101.

[26] CHEN, J., AND SAYED, A. Diffusion Adaptation Strategies for Distributed Optimization and Learning Over Networks. *IEEE Trans. Signal Processing 60*, 8 (2012), 4289–4305.

[27] CHOI, H. C., AND AHN, B. H. Image alignment by parameter hypersurface learning. *Electronics Letters 52*, 18 (2016), 1526–1528.

[28] CIANCIO, A., PATTEM, S., ORTEGA, A., AND KRISHNAMACHARI, B. Energy-efficient data representation and routing for wireless sensor networks based on a distributed wavelet compression algorithm. In *Proc. IEEE/ACM Symp. Inf. Process. Sensor Networks* (2006), pp. 309–316.

[29] COMBETTES, P. L., AND PESQUET, J.-C. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*. Springer, 2011, pp. 185–212.

[30] CROCCO, M., BUE, A. D., AND MURINO, V. A bilinear approach to the position self-calibration of multiple sensors. *IEEE Trans Signal Process. 60*, 2 (2012), 660–673.

[31] DIMAKIS, A. G., KAR, S., MOURA, J. M. F., RABBAT, M. G., AND SCAGLIONE, A. Gossip Algorithms for Distributed Signal Processing. *Proceedings of the IEEE 98*, 11 (2010), 1847–1864.

[32] DUCHI, J., AGARWAL, A., AND WAINWRIGHT, M. J. Dual Averaging for Distributed Optimization: Convergence Analysis and Network Scaling. In *IEEE Trans. Automatic Control* (2012), vol. 57, pp. 592–606.

[33] ECKSTEIN, J., AND FERRIS, M. C. Some reformulations and applications of the alternating direction method of multipliers. *Large Scale Optimization: State of the Art* (1993), 119–138.

[34] ERSEGHE, T. Distributed optimal power flow using ADMM. *IEEE transactions on power systems 29*, 5 (2014), 2370–2380.

[35] EVERETT, H. Generalized lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research 11* (1963).

[36] FORTIN, M., AND GLOWINSKI, R. On decomposition-coordination methods using an augmented lagrangian. *Augmented Lagrangian Methods: Applications to the Solution of Boundary-Value Problems* (1983).

[37] FRAMPTON, K. Acoustic self-localization in a distributed sensor network. *Sensors Journal, IEEE 6*, 1 (Feb 2006), 166–172.

[38] FRIEDMAN, J., HASTIE, T., AND TIBSHIRANI, R. A note on the group lasso and a sparse group lasso. *arXiv preprint arXiv:1001.0736* (2010).

[39] FUKUSHIMA, M. Application of the alternating direction method of multipliers to separable convex programming problems. *Computational Optimization and Applications 1* (1992), 93–111.

[40] GABAY, D. Applications of the method of multipliers to variational inequalities. *Augmented Lagrangian Methods: Applications to the Solution of Boundary-Value Problems* (1983).

[41] GABAY, D., AND MERCIER, B. A dual algorithm for the solution of nonlinear variational problems via finite element approximations. *Computers and Mathematics with Applications 2* (1976), 17–40.

[42] GALLAGER, R. G. *Low Density Parity Check Codes*. M.I.T. Press, 1963.

[43] GANNOT, S., BURSHTEIN, D., AND WEINSTEN, E. Signal Enhancement using Beamforming and Nonstationarity with Applications to Speech. *IEEE Transactions on Signal Processing* (2001).

[44] GARRETT, M. A. Radio astronomy transformed: Aperture arrays; past, present and future. In *AFRICON, 2013* (Sept 2013), pp. 1–5.

[45] GAUBITCH, N., KLEIJN, W. B., AND HEUSDENS, R. Auto-localization in ad-hoc microphone arrays. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on* (May 2013), pp. 106–110.

[46] GEOFFRION, A. M. Generalized benders decomposition. *Journal of Optimization Theory and Applications 10* (1972).

[47] GERSHMAN, A. Robust adaptive beamforming: an overview of recent trends and advances in the field. In *Antenna Theory and Techniques, 2003. 4th International Conference on* (Sept 2003), vol. 1, pp. 30–35 vol.1.

[48] GLOWINSKI, R., AND MARROCCO, A. Sur l'approximation, par elements finis d'ordre un, et la resolution, par penalisation-dualit'e, d'une classe de problems de dirichlet non lineares. *Revue Fran¸caise d'Automatique, Informatique, et Recherche Op'erationelle 9* (1975), 41–76.

[49] GOLUB, G. H., HANSEN, P. C., AND O'LEARY, D. P. Tikhonov regularization and total least squares. *SIAM Journal on Matrix Analysis and Applications 21*, 1 (1999), 185–194.

[50] GOLUB, G. H., AND VAN LOAN, C. F. *Matrix computations*, vol. 3. JHU Press, 2012.

[51] GULREZ, T., AND KAVAKLI, M. Precision position tracking in virtual reality environments using sensor networks. In *2007 IEEE International Symposium on Industrial Electronics* (June 2007), pp. 1997–2003.

[52] HAGHIGHAT, M. B. A., AGHAGOLZADEH, A., AND SEYEDARABI, H. A non-reference image fusion metric based on mutual information of image features. *Computers and Electrical Engineering 37* (2011).

[53] HAJINEZHAD, D., AND HONG, M. Nonconvex alternating direction method of multipliers for distributed sparse principal component analysis. In *IEEE Global Conference on Signal and Information Processing (GlobalSIP)* (2015).

[54] HARARY, F. *Graph Theory*. Addison-Wesley, Reading, MA, 1994.

[55] HAYKIN, S. *Adaptive Filter Theory*. Prentice Hall, NJ, 1996.

[56] HENDRIKS, R. C., HEUSDENS, R., AND JENSEN, J. MMSE Based Noise PSD Tracking with Low Complexity. *IEEE International Conference on Acoustics, Speech and Signal Processing* (2010).

[57] HESTENES, M. R. Multiplier and gradient methods. *Journal of Optimization Theory and Applications 4* (1969), 302–320.

[58] HESTENES, M. R. Multiplier and gradient methods. *Computing Methods in Optimization Problems* (1969).

[59] HEUSDENS, R., ZHANG, G., HENDRIKS, R. C., ZENG, Y., AND KLEIJN, W. B. Distributed MVDR beamforming for (wireless) microphone networks using message passing. In *Acoustic Signal Enhancement; Proceedings of IWAENC 2012; International Workshop on* (Sept 2012), pp. 1–4.

[60] HOLLINGER, G. A., YERRAMALLI, S., SINGH, S., MITRA, U., AND SUKHATME, G. S. Distributed data fusion for multirobot search. *IEEE Transactions on Robotics 31*, 1 (Feb 2015), 55–66.

[61] HONG, M., HAJINEZHAD, D., AND ZHAO, M.-M. Prox-PDA: The Proximal Primal-Dual Algorithm for Fast Distributed Nonconvex

Optimization and Learning Over Networks. In *International Conference on Machine Learning (ICML)* (2017).

[62] HSU, K.-J., LIN, Y.-Y., AND CHUANG, Y.-Y. Robust image alignment with multiple feature descriptors and matching-guided neighborhoods. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2015), pp. 1921–1930.

[63] HUNTER, A., OWENS, J., AND CARPENTER, M. A neural system for automated cctv surveillance. In *Intelligence Distributed Surveillance Systems, IEE Symposium on (Ref. No. 2003/10062)* (Feb 2003), pp. 14/1–14/5.

[64] IUTZELER, F., BIANCHI, P., CIBLAT, P., AND HACHEM, W. Asynchronous distributed optimization using a randomized alternating direction method of multipliers. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on* (Dec 2013), pp. 3671–3676.

[65] IUTZELER, F., BIANCHI, P., CIBLAT, P., AND HACHEM, W. Linear convergence rate for distributed optimization with the alternating direction method of multipliers. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on* (Dec 2014), pp. 5046–5051.

[66] JAKOVETIC, D., XAVIER, J., AND MOURA, J. Convergence rate analysis of distributed gradient methods for smooth optimization. In *Telecommunications Forum (TELFOR), 2012 20th* (Nov 2012), pp. 867–870.

[67] JAKOVETIC, D., XAVIER, J., AND MOURA, J. M. Fast distributed gradient methods. *IEEE Transactions on Automatic Control 59*, 5 (2014), 1131–1146.

[68] JIANG, F., KUANG, Y., AND ASTROM, K. Time delay estimation for TDOA self-calibration using truncated nuclear norm regularization.

In *Proc. IEEE Intl. Conf. on Acoust., Speech, Signal Process. (ICASSP)* (2013), pp. 3885–3889.

[69] JONES, D. L. Technology challenges for the square kilometer array. *IEEE Aerospace and Electronic Systems Magazine 28*, 2 (Feb 2013), 18–23.

[70] JONES, G., HARDING, C., AND LEUNG, V. Fusion of data from visual and low-resolution thermal cameras for surveillance. In *Intelligence Distributed Surveillance Systems, IEE Symposium on (Ref. No. 2003/10062)* (Feb 2003), pp. 17/1–17/5.

[71] JOSHI, S., CODREANU, M., AND LATVA-AHO, M. Distributed SINR balancing for MISO downlink systems via the alternating direction method of multipliers. In *Proceedings of 11th International Symposium on Modeling & Optimization in Mobile, Ad Hoc & Wireless Networks (WiOpt)* (May 2013), pp. 318–325.

[72] KANTAS, N., SINGH, S., AND DOUCET, A. Distributed maximum likelihood for simultaneous self-localization and tracking in sensor networks. *Signal Processing, IEEE Transactions on 60*, 10 (Oct 2012), 5038–5047.

[73] KAR, S., TANDON, R., POOR, H. V., AND CUI, S. Distributed detection in noisy sensor networks. In *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on* (2011), IEEE, pp. 2856–2860.

[74] KLUCKNER, S., POCK, T., AND BISCHOF, H. *Exploiting Redundancy for Aerial Image Fusion Using Convex Optimization*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 303–312.

[75] KSCHISCHANG, F., FREY, B., AND LOELIGER, H.-A. Factor graphs and the sum-product algorithm. *Information Theory, IEEE Transactions on 47*, 2 (Feb 2001), 498–519.

[76] LANG, S. *Complex Analysis*, 3 ed. New York: Springer-Verlag, 1993.

[77] LASDON, L. S. *Optimization Theory for Large Systems*. MacMillan, 1970.

[78] LEE, H., BATTLE, A., RAINA, R., AND NG, A. Y. Efficient sparse coding algorithms. In *Advances in neural information processing systems* (2006), pp. 801–808.

[79] LEE, S., KWON, H., AND SHIN, V. Distributed fusion filter on images with time delays. In *Computer Graphics, Imaging and Visualization (CGIV), 2011 Eighth International Conference on* (Aug 2011), pp. 98–102.

[80] LEVINSON, N. The wiener rms (root-mean-square) error criterion in filter design and prediction. *Journal of Mathematical Physics 25* (January 1947), 261–278.

[81] LI, M., ZHENG, G., AND LI, J. Clock self-synchronization protocol based on distributed diffusion for wireless sensor networks. *International Journal of Future Generation Communication and Networking 7* (2014).

[82] LING, Q., SHI, W., WU, G., AND RIBEIRO, A. DLM: Decentralized linearized alternating direction method of multipliers. *Signal Processing, IEEE Transactions on PP*, 99 (2015), 1–1.

[83] LIU, D. C., AND NOCEDAL, J. On the limited memory bfgs method for large scale optimization. *Mathematical programming 45*, 1 (1989), 503–528.

[84] LIU, Y., HU, Y. H., AND PAN, Q. Distributed, robust acoustic source localization in a wireless sensor network. *Signal Processing, IEEE Transactions on 60*, 8 (Aug 2012), 4350–4359.

[85] LORENZO, P. D., AND SCUTARI, G. NEXT: In-Network Nonconvex Optimization. *IEEE Transactions on Signal and Information Processing over Networks 2*, 2 (2016), 120–136.

[86] LUTZELER, F., CIBLAT, P., AND HACHEM, W. Analysis of Sum-Weight-Like Algorithms for Averaging in Wireless Sensor Networks. *IEEE Trans. Signal Processing 61*, 11 (2013), 2802–2814.

[87] MARKOVICH-GOLAN, S., BERTRAND, A., MOONEN, M., AND GANNOT, S. Optimal distributed minimum-variance beamforming approaches for speech enhancement in wireless acoustic sensor networks. *Signal Processing 107* (2015), 4–20.

[88] MEIER, L., VAN DE GEER, S., AND BÜHLMANN, P. The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 70*, 1 (2008), 53–71.

[89] MESSERSCHMITT, D. G. Stationary points of a real-valued function of a complex variable. Tech. Rep. UCB/EECS-2006-93, EECS Department, University of California, Berkeley, Jun 2006.

[90] MOALLEMI, C. C., AND ROY, B. V. Convergence of Min-Sum Message Passing for Quadratic Optimization. *IEEE Trans. Inf. Theory 55*, 5 (2009), 2413–2423.

[91] MOALLEMI, C. C., AND ROY, B. V. Convergence of Min-Sum Message Passing for Convex Optimization. *IEEE Trans. Inf. Theory 56*, 4 (2010), 2041–2050.

[92] MOKHTARI, A., SHI, W., LING, Q., AND RIBEIRO, A. DQM: Decentralized quadratically approximated alternating direction method of multipliers. *IEEE Transactions on Signal Processing 64*, 19 (Oct 2016), 5158–5173.

[93] MONTANARI, A., PRABHAKAR, B., AND TSE, D. Belief Propagation Based Multi-User Detection. In *Proc. 43rd Allerton Conf. on Communications, Control and Computing* (2005).

[94] MOREAU, J.-J. Fonctions convexes duales et points proximaux dans un espace hilbertien. *CR Acad. Sci. Paris Sér. A Math 255* (1962), 2897–2899.

[95] MÉZARD, M., AND MONTANARI, A. *Information, Physics, and Computation*. OUP Oxford, 2009.

[96] NEDIĆ, A., AND OZDAGLAR, A. Distributed Subgradient Methods for Multi-agent Optimization. *IEEE Transactions on Automatic Control* (2008).

[97] O'CONNOR, M., AND KLEIJN, W. B. Diffusion-Based Distributed MVDR Beamformer. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on* (May 2014), pp. 810–814.

[98] O'CONNOR, M., KLEIJN, W. B., AND ABHAYAPALA, T. Distributed sparse MVDR beamforming using the bi-alternating direction method of multipliers. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2016), IEEE, pp. 106–110.

[99] O'CONNOR, M., KLEIJN, W. B., AND ABHAYAPALA, T. Distributed TV-L1 Image Fusion Using PDMM. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (2017), pp. 3326–3330.

[100] O'CONNOR, M., ZHANG, G., KLEIJN, W. B., AND ABHAYAPALA, T. Function Splitting and Quadratic Approximation of the Primal-Dual Method of Multipliers for Distributed Optimization over Graphs.

submitted to IEEE. Trans. Signal and Information Processing over Networks, 2017.

[101] OMAR, Z., AND STATHAKI, T. Image fusion: An overview. *International Conference on Intelligent Systems, Modelling and Simulations 5* (2014).

[102] PALOMAR, D. P., AND CHIANG, M. Alternative distributed algorithms for network utility maximization: Framework and applications. *IEEE Transactions on Automatic Control 52*, 12 (2007), 2254–2269.

[103] PARIKH, N., AND BOYD, S. P. Proximal algorithms. *Foundations and Trends in optimization 1*, 3 (2014), 127–239.

[104] PEARL, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[105] PURE, A. A., GUPTA, N., AND SHRIVASTAVA, M. An overview of different image fusion methods for medical applications. *International Journal of Scientific and Engineering Research 4* (2013).

[106] QIU, W., HAO, P., AND SKAFIDAS, E. Distributed source localization in wireless sensor networks. In *Communications, Circuits and Systems (ICCCAS), 2010 International Conference on* (July 2010), pp. 90–94.

[107] RABBAT, M., AND NOWAK, R. Distributed optimization in sensor networks. In *Proceedings of the 3rd international symposium on Information processing in sensor networks* (2004), ACM, pp. 20–27.

[108] RATNASAMY, S., KARP, B., SHENKER, S., ESTRIN, D., GOVINDAN, R., YIN, L., AND YU, F. Data-centric storage in sensornets with GHT, a geographic hash table. *Mobile Networks and Applications 8*, 4 (2003), 427–442.

[109] RICHTÁRIK, P., AND TAKÁČ, M. Distributed coordinate descent method for learning with big data. *arXiv preprint arXiv:1310.2059* (2013).

[110] RUSMEVICHIENTONG, P., AND ROY, B. B. An analysis of Belief Propagation on the Turbo Decoding Graph with Gaussian Densities. *IEEE Trans. Inf. Theory 47*, 2 (2001), 745–765.

[111] S. BARMAN AND X. LIU AND S. DRAPER AND B. RECHT. Decomposition Methods for Large Scale LP Decoding. arXiv:1204.0556 [cs.IT], 2012.

[112] SAFAVI, S., AND KHAN, U. A. Revisiting Finite-Time Distributed Algorithms via Successive Nulling of Eigenvalue. *IEEE Signal Processing Letters 22*, 1 (2015), 54–57.

[113] SANDRYHAILA, A., KAR, S., AND MOURA, J. M. F. Finite-Time Distributed Consensus Through Graph Filters. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (2014), pp. 1080–1084.

[114] SAYED, A. H. Diffusion Adaptation over Networks. *E-Reference Signal Processing* (2013).

[115] SCHMALENSTROEER, J., JEBRAMCIK, P., AND HAEB-UMBACH, R. A gossiping approach to sampling clock synchronization in wireless acoustic sensor networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on* (May 2014), pp. 7575–7579.

[116] SCHMIDT, M., FUNG, G., AND ROSALES, R. Fast optimization methods for l_1 regularization: A comparative study and two new approaches. In *Machine Learning: ECML 2007*. Springer, 2007, pp. 286–297.

[117] SCHMIDT, M., FUNG, G., AND ROSALES, R. Optimization methods for l1-regularization. *University of British Columbia, Technical Report TR-2009 19* (2009).

[118] SEDGEWICK, R., AND WAYNE, K. *Algorithms*, 4th ed. Addison-Wesley Professional, 2011.

[119] SHEN, C., CHANG, T.-H., WANG, K.-Y., QIU, Z., AND CHI, C.-Y. Distributed robust multicell coordinated beamforming with imperfect CSI: An ADMM approach. *IEEE Transactions on signal processing 60*, 6 (2012), 2988–3003.

[120] SHENTAL, O., SIEGEL, P., WOLF, J., BICKSON, D., AND DOLEV, D. Gaussian belief propagation solver for systems of linear equations. In *Information Theory, 2008. ISIT 2008. IEEE International Symposium on* (July 2008), pp. 1863–1867.

[121] SHERSON, T., KLEIJN, W. B., AND HEUSDENS, R. A Distributed Algorithm for Robust LCMV Beamforming. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (2016), pp. 101–105.

[122] SHI, H., TIAN, B., AND WANG, Y. Fusion of multispectral and panchromatic satellite images using principal component analysis and nonsubsampled contourlet transform. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2010 Seventh International Conference on* (Aug 2010), vol. 5, pp. 2312–2315.

[123] SHI, W., LING, Q., WU, G., AND YIN, W. Extra: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization 25* (2014), 944–966.

[124] SIPSER, M. *Introduction to the Theory of Computation*. Course Technology Inc., 2006.

[125] SORBER, L., VAN BAREL, M., AND DE LATHAUWER, L. Unconstrained optimization of real functions in complex variables. *Society for Industrial and Applied Mathematics 22* (2012).

[126] SRIVASTAVA, R., PRAKASH, O., AND KHARE, A. Local energy-based multimodal medical image fusion in curvelet domain. *IET Computer Vision 10*, 6 (2016), 513–527.

[127] T. SHERSON AND R. HEUSDENS, W. B. KLEIJN. Derivation and analysis of the primal-dual method of multipliers based on monotone operator theory. arXiv:1706.02654 [math.OC], 2017.

[128] TASESKA, M., MARKOVICH-GOLAN, S., HABETS, E., AND GANNOT, S. Near-field source extraction using speech presence probabilities for ad hoc microphone arrays. In *Acoustic Signal Enhancement (IWAENC), 2014 14th International Workshop on* (Sept 2014), pp. 169–173.

[129] TAVAKOLI, V., JENSEN, J., CHRISTENSEN, M., AND BENESTY, J. Pseudo-coherence-based mvdr beamformer for speech enhancement with ad hoc microphone arrays. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on* (April 2015), pp. 2659–2663.

[130] TAVAKOLI, V. M., JENSEN, J. R., HEUSDENS, R., BENESTY, J., AND CHRISTENSEN, M. G. Ad hoc microphone array beamforming using the primal-dual method of multipliers. In *Proc. Euro- pean Signal Processing Conf.* (2016), pp. 1088–1092.

[131] TAVAKOLI, V. M., JENSEN, J. R., HEUSDENS, R., BENESTY, J., AND CHRISTENSEN, M. G. Distributed max-SINR speech enhancement with ad hoc microphone arrays. In *ICASSP* (2017), pp. 151–155.

[132] UHER, J., WYSOCKI, T. A., AND WYSOCKI, B. J. Review of Distributed Beamforming. *University of Nebraska-Lincoln, Omaha, USA* (2011).

[133] VEEN, B. D. V., AND BUCKLEY, K. M. Beamforming: A Versatile Approach to Spatial Filtering. *IEEE ASSP Magazine* (1988).

[134] WAINWRIGHT, M., AND JORDAN, M. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning 1(1-2)* (2008), 1–305.

[135] WANG, Q., HEMPSTEAD, M., AND YANG, W. A realistic power consumption model for wireless sensor network devices. In *2006 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks* (Sept 2006), vol. 1, pp. 286–295.

[136] WANG, Q., YU, D., AND SHEN, Y. An overview of image fusion metrics. In *Instrumentation and Measurement Technology Conference, 2009. I2MTC '09. IEEE* (May 2009), pp. 918–923.

[137] WANG, Z., ZIOU, D., ARMENAKIS, C., LI, D., AND LI, Q. A comparative analysis of image fusion methods. *IEEE Trans. Geosci. Remote Sens* (2005).

[138] WEI, E., AND OZDAGLAR, A. On the O (1/k) convergence of asynchronous distributed alternating direction method of multipliers. In *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE* (2013), IEEE, pp. 551–554.

[139] WEISS, Y., AND FREEMAN, W. T. Correctness of belief propagation in gaussian graphical models of arbitrary topology. *Neural Computation 13* (2001), 2173–2200.

[140] WIENER, N. *Extrapolation, Interpolation and Smoothing of Stationary Time Series*. New York: John Wiley and Sons, 1949.

[141] WIENER, N., AND HOPF, E. On a class of singular integral equations. *Proc. Prussian Acad., Math.-Phys. Ser.* (1931), 696.

[142] WIESEL, A., AND HERO, A. Distributed covariance estimation in gaussian graphical models. *Signal Processing, IEEE Transactions on 60*, 1 (Jan 2012), 211–220.

[143] WU, Z. Y., AND KHALIEFA, M. *Cloud Computing for High Performance Optimization of Water Distribution Systems*. ch. 71, pp. 679–686.

[144] XIE, Q. W., HE, J. C., QIAN, L., MITA, S., CHEN, X., AND JIANG, A. Image fusion based on tv-l1 function. In *2013 International Conference on Wavelet Analysis and Pattern Recognition* (July 2013), pp. 173–177.

[145] YEDIDIA, J., FREEMAN, W., AND WEISS, Y. Understanding belief propagation and its generalizations. In *Exploring Artificial Intelligence in the New Millennium*, G. Lakemeyer and B. Nebel, Eds. Morgan Kaufmann Publishers, Jan. 2003, ch. 8, pp. 239–236.

[146] YUAN, D., XU, S., AND ZHAO, H. Distributed primal-dual subgradient method for multiagent optimization via consensus algorithms. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 41*, 6 (Dec 2011), 1715–1724.

[147] YUAN, J., MILES, B., GARVIN, G., TAI, X.-C., AND FENSTER, A. Efficient convex optimization approaches to variational image fusion. *Numerical Mathematics: Theory, Methods and Applications 7*, 2 (May 2015), 234–250.

[148] YUAN, K., LING, Q., AND YIN, W. On the convergence of decentralized gradient descent. *SIAM Journal on Optimization 26*, 3 (2016), 1835–1854.

[149] ZENG, Y., AND HENDRIKS, R. C. Distributed delay and sum beamformer for speech enhancement in wireless sensor networks via ran-

domized gossip. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on* (2012), IEEE, pp. 4037–4040.

[150] ZENG, Y., AND HENDRIKS, R. C. Distributed Delay and Sum Beamformer for Speech Enhancement via Randomized Gossip. *IEEE/ACM Trans. Audio, Speech and Language Processing 22*, 1 (2014), 260–273.

[151] ZENG, Y., AND HENDRIKS, R. C. Distributed estimation of the inverse of the correlation matrix for privacy preserving beamforming. *Signal Processing 107* (2015), 109–122.

[152] ZHANG, G., AND HEUSDENS, R. Generalized linear coordinate-descent message-passing for convex optimization. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on* (March 2012), pp. 2009–2012.

[153] ZHANG, G., AND HEUSDENS, R. Bi-alternating direction method of multipliers. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on* (2013), IEEE, pp. 3317–3321.

[154] ZHANG, G., AND HEUSDENS, R. Bi-alternating direction method of multipliers. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on* (May 2013), pp. 3317–3321.

[155] ZHANG, G., AND HEUSDENS, R. Bi-alternating direction method of multipliers over graphs. *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on* (April 2015).

[156] ZHANG, G., AND HEUSDENS, R. On Simplifying the Primal-Dual Method of Multipliers. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (March 2016), pp. 4826–4830.

[157] ZHANG, G., AND HEUSDENS, R. Distributed Optimization using the Primal-Dual Method of Multipliers. accepted by IEEE Trans. Signal and Information Processing over Networks, 2017.

[158] ZHANG, G., AND HEUSDENS, R. Distributed optimization using the primal-dual method of multipliers. *IEEE Transactions on Signal and Information Processing over Networks* (2017).

[159] ZHANG, G., HEUSDENS, R., AND KLEIJN, W. On the convergence rate of the bi-alternating direction method of multipliers. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on* (May 2014), pp. 3869–3873.

[160] ZHANG, G., HEUSDENS, R., AND KLEIJN, W. B. Large Scale LP Decoding with Low Complexity. *IEEE Communications Letters 17*, 11 (2013), 2152–2155.

[161] ZHANG, G., KLEIJN, W. B., AND HEUSDENS, R. On Relationship between Primal-Dual Method of Multipliers and Kalman Filter. arXiv:1708.06881 [math.OC], 2017.

[162] ZHANG, H. M. Distributed Convex Optimization: A Study on the Primal-Dual Method of Multipliers. Master's thesis, Delft University of Technology, 2015.

[163] ZOU, H., AND HASTIE, T. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 67*, 2 (2005), 301–320.