Convex Optimization for Distributed Acoustic Beamforming

by

Dayle Raymond Jellyman

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the
requirements for the degree of
Master of Engineering
in Electronics and Computer Systems Engineering.

Victoria University of Wellington
2017

Abstract

Beamforming filter optimization can be performed over a distributed wireless sensor network, but the output calculation remains either centralized or linked in time to the weights optimization. We propose a distributed method for calculating the beamformer output which is independent of the filter optimization. The new method trades a small decrease in signal to noise performance for a large decrease in transmission power. Background is given on distributed convex optimization and acoustic beamforming. The new model is described with analysis of its behaviour under independent noise. Simulation results demonstrate the desirable properties of the new model in comparison with centralized output computation.

Acknowledgments

Thanks Kate, Bastiaan and David for your help and patience.

List of Abbreviations

AD-ADMM	Asynchronous distributed alternating direction method of multipliers
ADMM	Alternating direction method of multipliers
BFFW	Beamfomer filter weights
BFOP	Beamformer output
Bi-ADMM	Bi-alternating direction method of multipliers
D-ADMM	Distributed alternating direction method of multipliers
DANSE	Distributed adaptive node specific signal estimation
DC	Direct current
DFT	Discrete Fourier transform
FFT	Fast Fourier transform
GLiCD	Generalized linear coordinate descent
GSC	Generalized sidelobe canceller
KKT	Karush Kuhn Tucker
LASSO	Least absolute shrinkage and selection operator
LCMV	Linearly constrained minimum variance
MAP	Maximum a posteriori
MMSE	Minimum mean squared error
MVDR	Minimum variance distortionless response
PDMM	Primal dual method of multipliers
PSN	Public sensor network
SMI	Sample matrix inversion
SNR	Signal to noise ratio
STFT	Short time Fourier transform

Contents

Chapter 1

Introduction

Electronic sensors are an integral part of modern life. We include several types of sensors in our phones, drive on smart motorways, even washing machines know when they have been loaded unevenly. It has become normal for an individual to have tens, hundreds, even thousands of sensors in their service.

Every sensor and the data it generates is tainted by noise and error. Distinguishing between desired and undesired signals can be difficult, many clever ideas and systems have been developed to address this problem. The general aim of these systems is to increase the energy in the signal that we want, relative to the energy in the signal that we do not want, that is, an increase in signal to noise ratio (SNR).

Using multiple sensors is one method for increasing the SNR. When two sensors make independent observations of a target signal, both of the observations also contain noise. The observations of the target signal are correlated so phase aligning and then summing the two observations causes the target signal to sum constructively. Independent noise in the observations is uncorrelated so is summed non-constructively. The energy in the desired signal has increased while the energy in the noise has not. An increase in SNR has been achieved simply by summing the two phase aligned sensor signals.

If summing two sensors can increase the SNR, it is natural to want to sum ten sensors, or one billion sensors, or an unlimited number. However, more sensors means more data which requires more processing power. Increasing the processing power could mean using a more powerful processor, but this is still limited by manufacturing and fundamental constraints. The other option is to increase the processing power by using more processors. Using multiple spatially separated processors in concert is called distributed processing and is the central topic of this thesis.

Distributed processing requires a different type of algorithm from centralized processing. Each processor in the distributed network is given a local algorithm to run, communicates with neighboring processors, and by their combined efforts produce a global behaviour. Perhaps a useful analogy is to a flock of birds where each bird follows a local set of rules: do not crash and do not leave the flock, but knows nothing of the meta behaviour. The beautiful swirling flocking behaviour we observe is the global result of local rules and local communication.

We envision the trend towards distributed computation growing and expect it to extend to the development of public sensor networks (PSNs). A PSN is a large scale network of sensors and processors that are provided, for example, by a central government for anyone to use. Applications could include a nationwide traffic monitoring network or pollution sensing. The future of massive sensor networks is discussed in [1], and [2] gives an overview of sensor networks and applications. In this thesis we deal specifically with acoustic beamforming on a wireless microphone sensor network.

In order to motivate this large scale wireless microphone network, imagine a network of nodes installed in the ceilings of buildings and homes across New Zealand. Each node possesses a microphone, a battery, a processor, and radio communication capability. Any specific node in the network is dormant for the majority of the time, and wakes up when the application requires it to. A user inserts an instruction at an insertion node,

the insertion node then recruits neighboring nodes as necessary for completion of the instruction. For example, Jane is at a conference having a conversation with James in a crowded room of talkers. Jane is having trouble understanding James. She instructs the public microphone network to isolate James' voice and return the audio to her, enabling her to easily follow the conversation. Multiple sensor techniques have been found to be a promising and realistic technology for hearing aid applications in [3].

Another conceivable application is hands free calling. Say Tom wants to call his sister Jo using the microphone network in his home. Tom walks freely around the house without a communication device, doing the vacuuming and listening to both Jo and music over the stereo. The network isolates Tom's voice as he moves around the house and Jo has no problem hearing or understanding Tom. Hopefully these examples illustrate that PSNs may be useful.

An ideal PSN ought to be infinitely scalable. As previously discussed, we cannot connect a nationwide sensor network to one centralized processor. There is too much data, and transmission times scale with distance slowing the calculation down. So we require localized distributed processing algorithms in order to realize PSNs. Distributed processing also has additional benefits to security, system robustness, power consumption, processing speed, is more likely to have a sensor near to the target, and is more likely to have line of sight access to the target.

Security: Whenever data is processed in a centralized manner, the single central processor sees all the data collected in the entire network. If someone were to gain unauthorized access to the processing node, they would also gain access to all of the network data. In a distributed network, the spread of the data is naturally limited. This gives distributed systems an inherent security advantage.

Robustness: If the processor in a centralized system fails, the entire network fails. Distributed processing makes robust node failure possible, where the failure of a node simply changes the network graph without

causing global failure of the system.

Power: Transmitting sensor data to a centralized processor can be expensive. In section 4.4 we show that distributed systems can require less power. Radio signals decay as r^{-2} where r is the distance from the transmitter (based on distribution of the energy over the surface of a sphere) so reducing transmission distances by using distributed processing can extend network lifetimes. In a realistic setting the decay with distance can be as bad as r^{-4} due to cancellation caused by reflections [1].

Speed: It takes more time to transmit data over longer distances, so reducing transmission distances has some benefit in processing speed. There are speed costs to distributed processing too so this benefit should be considered in more detail for specific applications.

Minimum distance: A single microphone is likely to be further from the target source than the closest microphone in an array [4]. As sound energy decays with distance but the noise floor does not, the nearest microphone is likely to have a higher SNR. Even if only the best microphone within the array is used, the array is likely to have a higher SNR than by using the single microphone [4].

Line of sight: Line of sight access to the target is more likely if we use an array rather than a single sensor. Line of sight observations will generally have shorter acoustic transmission distances so will have higher SNR. Room reflections also complicate the relationship between the observation and the target signal, so line of sight observation is preferable.

Having discussed some potential applications and benefits of distributed processing we now turn to the research and development that enables it. Our application is acoustic beamforming, which requires the optimization of a set of weights. The optimization tool we will use is convex optimization. Boyd and Vandenberghe released a comprehensive and highly regarded book on convex optimization and its application in 2004 [5]. They summarize the tools that allow us to derive optimal beamformer weights in a centralized manner, by forming the Lagrangian from the ob-

jective and the constraints and finding the optimum.

While [5] summarizes the tools for convex optimization, it does not extend to solving convex optimization problems over an arbitrary distributed network graph. Bertsekas and Tsitsiklis [6] gives a theoretical treatment of numerical distributed methods. Boyd then deals with applied distributed computation in Randomized Gossip Algorithms [7], with analysis of the distributed averaging problem under gossip constraints over an arbitrary network. Distributed averaging aims to compute the global average using only local computation and communication. In gossip algorithms a node asynchronously and randomly chooses a neighbor, the two nodes compute the average of their current estimates, which replaces both current estimates. Over time, the network converges to the global average, bounds on the convergence time are also given in [7]. See [8] for a general review of gossip algorithms, which shows that gossip algorithms can be applied to problems such as Kalman filtering.

The alternating direction method of multipliers (ADMM) was reviewed by Boyd, Parikh, Chu, Peleato, and Eckstein in [9]. ADMM is an algorithm which enables the distributed optimization of a convex problem. It is limited in that the primal update can be split across a maximum of two nodes, and the dual update requires global knowledge so must be computed centrally. ADMM is an iterative algorithm that alternates between solving a distributed primal minimization and a centralized dual gradient ascent step. The primal minimization is ideally analytically solvable but in many cases requires another layer of iterative algorithm to solve. Understanding ADMM requires us to introduce Dual Ascent, Dual Decomposition, the Augmented Lagrangian, and the Method of Multipliers [9].

The requirement of a central node and synchronous updating limit the application of ADMM. Fully distributed ADMM (D-ADMM) with no requirement for a central node was introduced in [10] with application to basis pursuit. The same authors also applied D-ADMM to general separable optimization problems in [11] including average consensus. D-ADMM

was shown to have low communication costs relative to competing state of the art algorithms. However, D-ADMM was still synchronous and affected by the slowest worker problem. Asynchronous distributed ADMM (AD-ADMM) was introduced and analyzed in [12, 13] and shown to have preferable convergence properties.

Based on ADMM but with better convergence rates, the primal dual method of multipliers (PDMM) was introduced by Zhang and Heusdens in [14] (it was originally called the bi-alternating direction method of multipliers (BiADMM)). PDMM minimizes the augmented bi-conjugate function and is bounded by 0, in contrast to ADMM which is a saddle point problem. They demonstrate the effectiveness of the algorithm with application to ℓ_1 regularized least squares minimization (LASSO).

With the addition of Kleijn, the same authors then analyze the convergence rate of PDMM in [15] and show that for general convex functions, PDMM converges in $\mathcal{O}(K^{-1})$. Results are given for application to LASSO showing that PDMM outperforms ADMM and fast ADMM. These papers had developed PDMM on two nodes only, Zhang and Heusdens then published PDMM for arbitrary graphs in [16]. The algorithm is capable of both asynchronous and synchronous implementation, and retains a convergence rate of $\mathcal{O}(K^{-1})$. Simplified update equations are presented in [17].

PDMM enables asynchronous distributed convex optimization on an arbitrary graph, and can be used to implement a minimum variance distortionless response (MVDR) beamformer. But what is a beamformer? What is MVDR? We now turn our attention to acoustic beamforming.

A beamformer is a method of increasing the SNR in an estimate of a target signal based on the target's location in space. The gain pattern sometimes looks like a beam, hence the name. Multiple sensors make observations from different locations, which are then delayed, weighted and summed to form the beamformer output (BFOP).

Many applications exist for beamforming, for example: towed sonar

arrays, cellphone networks, speech enhancement, radar, air traffic control, medical imaging, geo-physics, and astro-physics. Van Veen and Buckley give an overview of beamformers in [18] and give references for these applications.

Our interest is acoustic beamformers, so the sensors are microphones and the signals being observed are acoustic pressure waves. The distance from the sound source to each sensor is likely to be different, so the pressure wave arrives at each sensor at a different time. It is this arrival time information which enables beamforming. Delays are added to the sensor observations to phase align the target source across sensors. The delayed observations are then summed and the phase aligned target signal sums constructively. Any interfering source arriving from a different location will have a different set of delays associated with it and will not sum constructively. Any independent sensor noise present will also not be constructively summed. The result is the constructive summing of the target signal and non-constructive summing of interference signals and noise, increasing the SNR.

This type of beamformer is called delay and sum, or Bartlett, and was originally introduced for sonar and radar applications [19, 20]. The delay and sum beamformer is not optimal in the presence of interference or diffuse noise. Many optimal and adaptive beamformers have been subsequently proposed. Capon introduced a maximum likelihood filter [21] to suppress interference, which was optimal in a least squares sense, and was the first MVDR beamformer. Widrow proposed an adaptive least mean squares algorithm in [22] which required training on a reference signal and had a soft constraint. Griffiths [23] adaptively finds the minimum mean square error. Frost's 1972 algorithm was similar to Widrow but with hard constraints that were not restricted to unity, and with no requirement for a reference signal [24]. Frost's adaptive algorithm is capable of adapting without accumulating errors. He also gave a geometric analysis that is still instructive. The multiple sidelobe canceller was introduced by Ap-

plebaum in [25] but also required a reference signal. [26] gives a useful comparison of Applebaum, Widrow and Frost.

The beamformers described so far have all been centralized. Distributed beamformers became popular some forty years later. Zeng and Hendriks apply randomized gossip to delay and sum beamforming in [27] and achieve an asynchronous distributed beamformer. The approach was notable for placing no constraints on the network topology, however the noise must be uncorrelated i.e. no point source interferers or diffuse noise, and they require global averaging at each time sample. Heusdens, Zhang, Hendriks and Kleijn use message passing to achieve distributed MVDR beamforming in [28]. MVDR beamformers are able to null interference so are often preferable to delay and sum beamformers. An MVDR beamformer is reduced to delay and sum when the noise is uncorrelated.

The algorithm in [28] is named generalized linear coordinate descent (GLiCD) and requires a trade off parameter to control the relative weight of the diagonal vs off-diagonal components of the noise correlation matrix. GLiCD also requires a global averaging at each time sample, and for the covariance matrix sparsity pattern to match the network adjacency pattern, which requires adjusting the transmission range of nodes to match the covariance matrix. Bertrand and Moonen present distributed adaptive node specific signal estimation (DANSE) in [29], which is limited to a fully connected network. Any algorithm that requires a fully connected network is not infinitely scalable so can not be applied on a PSN. They optimize for minimum mean squared error (MMSE) and aim for a node specific estimation of the desired signal , that is, each node may estimate a different signal. DANSE is extended to LCMV beamforming (called LC-DANSE) in [30] but it again depends on a fully connected network and provides for node specific signal estimation. The advantage offered by this algorithm is reduced communication costs compared with the centralized LCMV beamformer case.

Bertand and Moonen present a distributed LCMV beamformer in [30]

that requires either a fully connected or acyclic network that results in the optimal output at every node. Markovich and Golan describe a distributed generalized sidelobe canceller (GSC) beamformer in [31] that also requires a fully connected network. These network configuration limitations rule these algorithms out for our application as we aim to implement a fully distributed beamformer filter weights (BFFW) optimization on adhoc and scalable networks. O'Connor and Kleijn propose a diffusion based beamformer that approximates MVDR in [32] that does not restrict the network topology explicitly. It does however transmit data that scales with the network size, excluding the possibility of an infinitely scalable network. Sherson, Kleijn and Heusdens introduce a novel distributed LCMV beamformer in [33] which can compute the optimal beamformer response on both cyclic and acyclic networks. The beamformer output calculation is connected to the weight calculation.

The application of PDMM to sparse distributed acoustic beamforming is presented by O'Connor, Kleijn and Abhayapala in [34]. They acknowledge that in a beamforming application on a large sensor network, it may not be necessary to involve all network nodes. Instead, their sparse beamformer uses an ℓ_1 norm regularization to encourage low weights to zero. All nodes are still involved in the BFFW optimization, but some nodes do not contribute to the BFOP. This raises the idea of optimization on a subnetwork rather than the full network generally considered in the literature.

Calculation of the BFFWs has received most of the attention in the literature, with less attention given to calculation of the BFOP. In [34], the BFOP is calculated in a centralized manner for demonstration of the algorithm. The optimal beamformer of [33] leaves open the choice of optimization algorithm so can also use PDMM, but the BFOP calculation is tied to the BFFW optimization. This could be viewed as a useful feature, however it is possible that there is some advantage to retaining separation as linking the calculations also links them in time. It is easy to imagine a

scenario where a system designer may prefer to run the two calculations at different rates. For example, in a static scenario the BFFWs need only be calculated once, while the BFOP is calculated constantly. Also, optimization of the BFFWs is an iterative and costly process, where the BFOP calculation can be computed explicitly. Given that we are working with a wireless sensor network where battery life is a primary concern, it makes sense to reduce the computational load where possible. What is missing is a method for the distributed calculation of the the BFOP that is independent of the BFFW optimization.

The primary contribution of this thesis is to propose a low cost method for distributed BFOP calculation that is independent of the BFFW optimization. Our proposed method requires that each node make a local estimate of the target signal, in contrast to centralized BFOP calculation where only the fusion node estimates the target signal. Each local estimate combines the local weighted observation and any estimates passed to it by upstream nodes (nodes further from the insertion node). The BFFWs are assumed to be already available, for example, as the result of the BFFW optimization described in [34]. Each node passes its local estimate to one neighboring downstream node only. This results in a tree structure for the BFOP calculation. As each node is only required to pass a single estimate to a neighboring node, the required transmission power is reduced compared to schemes that transmit the observations to a fusion node. The proposed distributed BFOP calculation is found to trade a small decrease in SNR performance for a large decrease in the required transmission power. This makes it possible to achieve a higher SNR at lower communication cost by using our new method.

The following chapter gives background details required for understanding and implementing distributed PDMM and acoustic beamforming. Chapter 3 provides some auxiliary results which are useful to anyone implementing a distributed acoustic beamformer, but which do not belong with the primary result. Included here is a result that suggests the optimal

distribution of nodes is not uniform. Chapter 4 gives the primary result of the thesis which describes the proposed distributed BFOP calculation and gives analysis and simulation results comparing its performance with centralized BFOP calculation. Discussion and suggestions for further development are then given. Chapter 5 gives conclusions and summarizes the thesis.

Chapter 2

Background

The introduction motivated distributed acoustic beamforming and summarized the related history and literature. Convex optimization was discussed through to the primal dual method of multipliers (PDMM), which is the current state of the art in distributed convex optimization. Beamforming was introduced and then combined with distributed optimization in distributed beamforming. This culminated in the PDMM based sparse distributed beamformer of [34] and the optimal beamformer of [33]. Both examples are capable of asynchronous distributed optimization of the beamformer filter weights (BFFWs) on an ad hoc network. However, the beamformer output (BFOP) calculation is left centralized in [34], and is distributed but linked to the BFFW optimization in [33]. This thesis proposes a method for distributed BFOP calculation that is independent of the BFFW optimization, and relies on a separate algorithm such as [34] to provide the BFFWs.

This chapter is intended to provide the specific details that enable implementation and understanding of the proposed distributed BFOP calculation and BFFW optimization using [34]. Convex optimization is developed from convexity through the method of Lagrange multipliers, iterative algorithms, and separable algorithms. The augmented Lagrangian then brings robustness but costs separability. ADMM gives us limited sep-

arability, then we arrive finally at PDMM over arbitrary graphs.

The second half of the chapter describes implementation and issues of practical acoustic beamformers. Discussion is given on the short time Fourier transform (STFT), conjugate symmetry for guaranteeing a real post filter time domain signal, conversion from the complex to real domain, and the gradient of a function of a complex variable. Beamforming details are then addressed including delay and sum, optimal MVDR, adaptive algorithms following Frost, finishing with the sparse distributed MVDR of [34].

2.1 Distributed Convex Optimization

The MVDR objective function is a quadratic [34] and therefore convex, so convex optimization can be used to find the optimal filter weights [35, 5]. Distributed convex optimization shifts the optimization from a central computation at a fusion centre, to being calculated over the sensor network. Implementation of distributed optimization requires an understanding of the building blocks, so this section introduces convexity and convex optimization, dual ascent, dual decomposition, ADMM, arriving finally at PDMM over arbitrary graphs.

2.1.1 Convexity

Convex optimization deals with optimizing a convex objective function under constraints. All details in this section are referenced in [5]. The standard form is

$$\begin{aligned} \text{minimize} \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq 0, \qquad i \in [1, 2, \ldots, m] \\ & g_i(x) = 0, \qquad i \in [1, 2, \ldots, n] \end{aligned} \tag{2.1}$$

where the objective function $f_0(x)$ is closed (contains all of its limit points, and contains its own boundary), proper (always greater than negative infinity, and less than positive infinity for at least one point on the domain), and convex. $f_i(x)$ is the ith convex inequality constraint, $g_i(x)$ is the ith affine equality constraint, $i \in \mathbb{Z}$, m is the number of inequality constraints, and n is the number of equality constraints.

A function f is convex if and only if the domain of f is convex, and for all x and y on the domain of f it satisfies Jensen's inequality,

$$\theta f(x) + (1-\theta)f(y) \geq f(\theta x + (1-\theta)y), \qquad 0 \leq \theta \leq 1. \tag{2.2}$$

Figure 2.1 gives an example. The chord between $f(x)$ and $f(y)$ is always greater than or equal to the function f between x and y. If the only place where the chord and the function touch is at the endpoints (arguments x and y) then the function is strictly convex and the inequality in equation (2.2) becomes strict. Convexity plays an important role in optimization, as a convex function has only one infimum which is globally optimal. Therefore, if you are able to find a point on a convex function where the gradient is zero, this is guaranteed to be globally optimal.

The first order Taylor approximation near x of a convex function $f(x)$ is a global underestimator of the function. That is, $f : \mathbb{R}^N \to \mathbb{R}$ is convex if and only if the domain of f is convex and,

$$f(y) \geq f(x) + \nabla f(x)^T (y - x), \tag{2.3}$$

where ∇ is the gradient operator. If the function is strictly convex then the first order Taylor approximation touches the function at one point only.

If f is twice differentiable, then it is convex if and only if the domain of f is convex and its Hessian is positive semidefinite for all x in the domain of f,

$$\nabla^2 f(x) \succcurlyeq 0, \qquad \forall x \in \mathbf{dom}\ f. \tag{2.4}$$

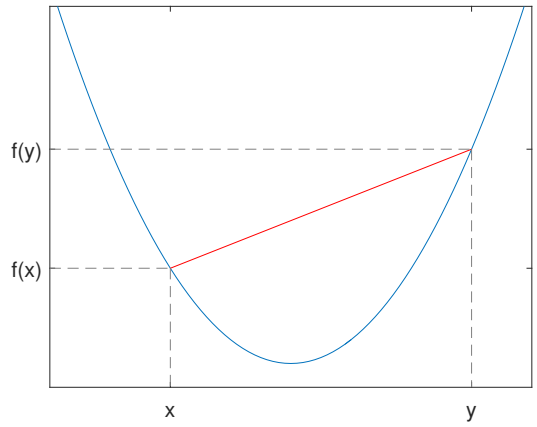

Figure 2.1: Visual interpretation of Jensen's inequality. The function f is strictly convex as the chord between any two points $(x, f(x))$ and $(y, f(y))$ always lies above f between x and y.

A symmetric matrix $A \in \mathbb{R}^{n \times n}$ is positive semidefinite if and only if it satisfies,

$$x^T A x \geq 0, \qquad \forall x \in \mathbb{R}^n. \tag{2.5}$$

A function that maps from the reals to the reals $f : \mathbb{R}^n \to \mathbb{R}$ can be convex and so too can $f : \mathbb{C}^n \to \mathbb{R}^m$ which maps from complex to real. A function $f : \mathbb{C}^n \to \mathbb{C}^m$ can not be convex. The complex numbers are unordered so it makes no sense to consider a complex number to be more or less than another.

The convex conjugate will play a role in some subsequent derivations and is given by,

$$f^*(y) = \sup_x \left\{ x^T y - f(x) \right\}. \tag{2.6}$$

The convex conjugate function has the desirable property of being always convex, as it is the pointwise supremum of affine functions [5]. There is no requirement for $f(x)$ to be convex or differentiable for this to be true. The convex conjugate (2.6) leads directly to Fenchel's inequality by observation,

$$f^*(y) \geq x^T y - f(x) \tag{2.7}$$

The general form of the convex conjugate is sometimes called the Fenchel conjugate. For a differentiable function the convex conjugate is sometimes referred to as the Legendre transform.

This section gave a brief introduction to convexity based on Boyd, for more detail see [5]. For a thorough and well regarded theoretical reference on convex analysis, see Rockafellar [36]. Having introduced convexity we need a tool for solving constrained convex problems of the form (2.1). That tool is Lagrange multipliers.

2.1.2 Lagrange Multipliers

Constrained optimization problems such as (2.1) require the constraints to be met exactly at all times. A useful tool is the method of Lagrange multipliers, which converts the problem from a constrained optimization to an unconstrained optimization with a penalty for violating the constraints. The method of Lagrange multipliers was introduced by Lagrange in 1788 [37] to determine general equations of equilibrium for problems with constraints [38]. We first introduce Lagrange multipliers $\lambda \in \mathbb{R}^m$ and reformulate the problem as a Lagrangian. For objective function $f_0(x)$ with $x \in \mathbb{R}^p$, inequality constraints $f_i(x)$, and equality constraints $g_i(x)$ the Lagrangian function $L : \mathbb{R}^p \times \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$ is defined as

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{n} \nu_i g_i(x). \tag{2.8}$$

λ_i is the Lagrange multiplier for the ith inequality constraint, ν_i is the Lagrange multiplier for the ith equality constraint. λ and ν are also called the dual variables. Minimization of the Lagrangian over the primal variable x gives the dual function,

$$g(\lambda, \nu) = \inf_x \left\{ f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{n} \nu_i g_i(x) \right\}. \tag{2.9}$$

Note that the dual function is a pointwise infimum of affine functions for fixed x and is therefore always concave [5]. This is true even when the original function is not convex. The maximization of the dual function is called the dual problem,

$$\begin{aligned} &\text{maximize} \;\; g(\lambda, \nu) \\ &\text{subject to} \;\; \lambda \succeq 0. \end{aligned} \tag{2.10}$$

The reason for our interest in the dual problem is that it is possible and sometimes simpler to solve (2.1) in the dual domain. If Slater's condition is met and the primal objective and constraints are convex then we have strong duality. Strong duality means that the optimum (maximum) of the dual problem is equal to the optimum (minimum) of the primal problem. Slater's condition requires that there exists feasible x such that the inequality constraints $f_i(x)$ are strictly less than zero,

$$f_i(x) < 0, \;\; \forall i. \tag{2.11}$$

If the inequality constraints are affine, then Slater's condition can be relaxed to

$$f_i(x) \leq 0. \tag{2.12}$$

If the objective function and constraints of the primal problem (2.1) are all convex, we can also use the Karush Kuhn Tucker (KKT) conditions to

prove optimality [5]. Specifically, if

$$\begin{aligned} f_i(x) &\leq 0, \qquad i \in [1, 2, \ldots, m] \\ g_i(x) &= 0, \qquad i \in [1, 2, \ldots, n] \\ \lambda_i &\geq 0, \qquad i \in [1, 2, \ldots, m] \\ \lambda_i f_i(x) &= 0, \qquad i \in [1, 2, \ldots, m] \\ \nabla f_0(x) + \sum_{i=1}^{m} \lambda_i \nabla f_i(x) + \sum_{i=1}^{n} \nu_i \nabla g_i(x) &= 0, \end{aligned} \tag{2.13}$$

then (x, λ) are the optimal primal and dual variables, and there is zero duality gap [5].

We now have the tools to identify a problem as convex, and to solve constrained convex optimization problems using Lagrange multipliers analytically. There are situations where an analytic solution is not available, and iterative methods must be used to find the optimum. In the next section we introduce dual ascent as an example of an algorithm that iteratively approaches the maximum of a function.

2.1.3 Iterative Algorithms: Dual Ascent

Often in optimization problems the solution is not available analytically. Iterative algorithms can be used to approach and approximate the solution. This section introduces iterative algorithms using dual ascent which iteratively finds the optimum of the dual function using gradient ascent. This review is based on dual ascent as it is presented in [9]. The problem statement is an equality constrained minimization,

$$\begin{aligned} &\text{minimize} \;\; f(x) \\ &\text{subject to} \;\; Ax = b. \end{aligned} \tag{2.14}$$

The Lagrangian is

$$L(x, \nu) = f(x) + \nu^T (Ax - b), \tag{2.15}$$

and the dual function $g(\nu)$ is given by,

$$\begin{aligned} g(\nu) &= \inf_x L(x, \nu) \\ &= \inf_x \Big\{ f(x) + \nu^T (Ax - b) \Big\} \\ &= -\nu^T b - \inf_x \Big\{ -\nu^T Ax - f(x) \Big\} \\ &= -\nu^T b - f^*(-A^T \nu), \end{aligned} \tag{2.16}$$

where f^* is the convex conjugate of f.

If we assume strong duality of f then we can use the maximizing argument of $g(\nu)$, ν^*, to retrieve the optimal value of the primal problem using

$$x^* = \arg\min_x L(x, \nu^*), \tag{2.17}$$

provided there is only one minimizer of $L(x, \nu^*)$, for example if $L(x, \nu^*)$ is strongly convex.

The dual ascent method alternates between gradient ascent on the dual variable to move towards the dual optimum, and updating the primal variable based on the new ν, as in (2.17). If we assume that $g(\nu)$ is differentiable, then the gradient of g is

$$\nabla g(\nu) = Ax - b, \tag{2.18}$$

(for intuition, this can be seen by fixing x in the third line of (2.16)). So we can step towards the optimal point by minimizing the primal function over $x^{(k)}$ to find $x^{(k+1)}$, where k is the iteration number, then stepping $\nu^{(k)}$ in the direction $Ax^{(k+1)} - b$, i.e. gradient ascent to find $\nu^{(k+1)}$. That

is, we alternately step the two variables towards the saddle point of the Lagrangian using the iterative updates,

$$\begin{aligned} x^{(k+1)} &:= \arg\min_x L(x, \nu^{(k)}) \\ \nu^{(k+1)} &:= \nu^{(k)} + \alpha^{(k)}(Ax^{(k+1)} - b). \end{aligned} \tag{2.19}$$

$\alpha^{(k)}$ is a step size to control the dual gradient step. As the dual variable gets closer to the optimum, the dual gradient becomes smaller. At the optimal point we have a fixed point as the maximum of the dual function implies that the dual gradient is zero, so the dual update simply returns the previous ν. No change in ν also implies no change in x.

We now have updates that will iterate towards the optimal point, as an example of how iterative algorithms such as PDMM work. However, we still do not have the updates explicitly. The objective function $f(x)$ is required in order to calculate the minimization in the x update. It is also possible that the x update step in (2.19) will require an additional iterative algorithm to perform the minimization. Several assumptions also have to be met, so the method can fail relatively easily. For example, if strong convexity is violated and $f(x)$ is an affine function with non zero slope then the minimization in the x update is unbounded below.

We have now seen an iterative algorithm for solving a convex optimization problem. Dual ascent is a centralized algorithm however, and we require a distributed algorithm. It is possible to separate the primal variable update across processors as long as the objective function and variable are separable. This method is called dual decomposition and is discussed in the following section as our introduction to distributed processing.

2.1.4 Separable Algorithms: Dual Decomposition

Dual ascent was able to iteratively approach the optimal point of the optimization problem in a centralized manner. We are aiming for a distributed algorithm, and dual ascent can be extended to be a partially distributed algorithm. If $f(x)$ is separable, that is,

$$f(x) = \sum_{i=1}^{n} f_i(x_i), \tag{2.20}$$

then the x update in (2.19) can be calculated in a distributed manner. Here $x = [x_1^T, x_2^T, \ldots, x_n^T]^T$, and n is the total number of partitions of x. Note that the ν update in (2.19) requires knowledge of all of $x^{(k+1)}$ so the dual variable update must still be calculated centrally. Nevertheless, this is our first distributable algorithm, and can be used to solve a problem in a distributed manner provided the objective function and variable are separable.

The matrix A must also be partitioned, in such a way that

$$Ax = \sum_{i=1}^{n} A_i x_i. \tag{2.21}$$

The Lagrangian (2.15) can now be rewritten as

$$L(x, \nu) = \sum_{i=1}^{n} L_i(x_i, \nu) = \sum_{i=1}^{n} \Big\{ f_i(x_i) + \nu^T A_i x_i - (1/n)\nu^T b \Big\}. \tag{2.22}$$

The x minimization can now be split across processors. The new update equations are,

$$\begin{aligned} x_i^{(k+1)} &:= \arg\min_{x_i} L_i(x_i, \nu^{(k)}) \\ \nu^{(k+1)} &:= \nu^{(k)} + \alpha^k (Ax^{(k+1)} - b). \end{aligned} \tag{2.23}$$

Dual ascent and dual decomposition require assumptions to be met, for example the strong convexity of f. These conditions are not guaranteed to be met in general, but are often met in our application. For example an MVDR beamformer has a strictly convex objective and linear constraints so strong duality only requires that the constraints are strictly met. It is also possible to ensure that the Lagrangian is strictly convex by the addition of a quadratic term, with the additional advantage of speeding up the optimization. The resulting function is called the augmented Lagrangian.

2.1.5 Augmented Lagrangian

The augmented Lagrangian guarantees strong convexity and speeds up convergence by adding a quadratic ℓ_2 norm term to the Lagrangian,

$$L_\rho(x, \nu) = f(x) + \nu^T(Ax - b) + \frac{\rho}{2}\|Ax - b\|_2^2, \tag{2.24}$$

where the definition of the p-norm is

$$\|x\|_p = \left(\sum_{i=1}^{N} |x_i|^p\right)^{1/p}, \qquad x \in \mathbb{R}^N. \tag{2.25}$$

The addition of the quadratic term results in a function which is always strictly convex, eliminating this limitation from the dual ascent and dual decomposition algorithms. The solution to L_ρ is equivalent to the solution to L, as when the constraint is met the quadratic term is zero. An iterative method for finding the optimal point can be found following the same reasoning as given above for the Lagrangian. This results in the following update equations,

$$\begin{aligned} x^{(k+1)} &:= \arg\min_x L_\rho(x, \nu^{(k)}) \\ \nu^{(k+1)} &:= \nu^{(k)} + \rho(Ax^{(k+1)} - b), \end{aligned} \tag{2.26}$$

and is known as the method of multipliers. The method of multipliers is more general than dual ascent, and has better convergence properties [9]. The cost is that the primal variable x is no longer separable as the quadratic introduced cross terms, so the primal update can no longer be distributed as it was in dual decomposition. However, the augmented Lagrangian can be made separable for distributed optimization using ADMM.

2.1.6 ADMM: Alternating Direction Method of Multipliers

Dual ascent can be used to find the optimum of a convex function, which leads to separability via dual decomposition allowing the primal update to be distributed across processors. Dual decomposition fails under some circumstances such as affine constraint functions, but can be made more robust by using the augmented Lagrangian and the method of multipliers. However, the penalty term in the augmented Lagrangian causes the method of multipliers to be inseparable, losing the ability for distributed processing. A relatively minor adjustment to the objective and constraint functions can make it possible to combine the augmented Lagrangian with separability. This method is called the alternating direction method of multipliers, or ADMM.

ADMM splits a splittable objective function and primal variable in two, so x becomes x and z, and $f(x)$ becomes $f(x) + g(z)$. We assume that the functions $f(x)$ and $g(z)$ are convex, and restate the equivalent problem as

$$\begin{aligned} \text{minimize} \quad & f(x) + g(z) \\ \text{subject to} \quad & Ax + Bz = c. \end{aligned} \tag{2.27}$$

The augmented Lagrangian is formed as before,

$$L_\rho(x, z, \nu) = f(x) + g(z) + \nu^T(Ax + Bz - c) + \frac{\rho}{2}\|Ax + Bz - c\|_2^2. \tag{2.28}$$

We now minimize over the two variables separately, then update the dual variable using gradient ascent,

$$\begin{aligned} x^{(k+1)} &:= \arg\min_{x} L(x, z^{(k)}, \nu^{(k)}) \\ z^{(k+1)} &:= \arg\min_{z} L(x^{(k+1)}, z, \nu^{(k)}) \\ \nu^{(k+1)} &:= \nu^{(k)} + \rho(Ax^{(k+1)} + Bz^{(k+1)} - c). \end{aligned} \tag{2.29}$$

By iterating over these updates we approach the optimal solution. The primal updates in ADMM are split across the two variables x and z, whereas if we applied dual ascent they would be minimized together. This accounts for alternating direction in the name of the algorithm.

ADMM is a useful and current algorithm that is competitive with other learning algorithms, sometimes outperforming state of the art in centralized computation [9]. The attractive feature of ADMM in our distributed beamformer context is the ability to calculate the updates in a distributed manner. Useful as this is, ADMM is limited in two important ways: 1) only the primal updates can be calculated in a distributed manner, the dual update still requires a centralized calculation, and 2) the objective and variable can only be split in two while still guaranteeing convergence [9]. An ideal distributed optimization algorithm would allow infinitely many primal variables and have no centralized update. Application of ADMM to basis pursuit is given in the following section, illustrating how a distributed algorithm can actually be used. We then move on to PDMM, the algorithm which we will use to implement fully distributed optimization of the BFFWs.

2.1.7 Basis Pursuit Using ADMM

This section gives an example application of ADMM to basis pursuit based on [9] in order to demonstrate how iterative distributed algorithms can be used in practice. Basis pursuit aims to find a basis for the representation

of data using less dimensions. It achieves that by minimizing the ℓ_1 norm which can produce sparse solutions [5]. The basis pursuit problem statement is

$$\begin{aligned} &\text{minimize} \;\; \|x\|_1 \\ &\text{subject to} \;\; Ax = b \end{aligned} \tag{2.30}$$

where $x \in \mathbb{R}^n, A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m, m < n$. Here b could represent observations of an underdetermined system of linear equations $b = Ax$ and we would like to find the solution x. An underdetermined system of equations typically has an infinite number of solutions (or possibly none), but imposing further restrictions such as requiring a sparse solution will favour a particular solution. In order to formulate the problem for ADMM, we create a split objective function that combines the ℓ_1 minimization and limits the space by putting it in consensus form,

$$\begin{aligned} &\text{minimize} \;\; f(x) + \|z\|_1 \\ &\text{subject to} \;\; x = z, \end{aligned} \tag{2.31}$$

where $f(x)$ is zero when $x \in \{x | Ax = b\}$ and infinity otherwise, i.e. $f(x)$ is an indicator function. The augmented Lagrangian for the problem is then

$$L_\rho(x, z, \nu) = f(x) + \|z\|_1 + \nu^T(x - z) + \frac{\rho}{2}\|x - z\|_2^2 \tag{2.32}$$

Now convert it to an equivalent scaled form by completing the square,

$$\begin{aligned} L_\rho(x, z, \nu) &= f(x) + \|z\|_1 + \frac{\rho}{2}\Big(\|x - z\|_2^2 + \frac{2}{\rho}\nu^T(x - z)\Big) \\ &= f(x) + \|z\|_1 + \frac{\rho}{2}\left(\left\|x - z + \frac{\nu}{\rho}\right\|_2^2 - \left(\frac{\nu}{\rho}\right)^2\right) \\ &= f(x) + \|z\|_1 + \frac{\rho}{2}\left\|x - z + \frac{\nu}{\rho}\right\|_2^2 - \frac{\nu^2}{2\rho} \end{aligned} \tag{2.33}$$

Minimizing $L_\rho(x, z, \nu)$ over the primal variables x and z separately, and then updating the dual variable ν using gradient ascent results in the ADMM updates,

$$\begin{aligned} x^{(k+1)} &:= \arg\min_x \; f(x) + \frac{\rho}{2}\Big\|x - z^{(k)} + \frac{\nu^{(k)}}{\rho}\Big\|_2^2 \\ z^{(k+1)} &:= \arg\min_z \; \|z\|_1 + \frac{\rho}{2}\Big\|x^{(k+1)} - z + \frac{\nu^{(k)}}{\rho}\Big\|_2^2 \\ \nu^{(k+1)} &:= \nu^{(k)} + x^{(k+1)} - z^{(k+1)}. \end{aligned} \tag{2.34}$$

The $f(x)$ term in the x update (2.34) restricts us to $\{x|Ax = b\}$. The quadratic term is a Euclidean projection of $z - \nu/\rho$ onto the set $\{x|Ax = b\}$ and is given by the solution to

$$\begin{aligned} &\text{minimize} \;\; \frac{1}{2}\big\|x - \big(z^k - (\nu^k/\rho)\big)\big\|_2^2 \\ &\text{subject to} \;\; Ax = b. \end{aligned} \tag{2.35}$$

The constrained minimization (2.35) problem can be converted to an unconstrained problem using another Lagrangian with multiplier y,

$$L(x, \nu) = \frac{1}{2}\big\|x - \big(z^k - (\nu^k/\rho)\big)\big\|_2^2 + y^T(Ax - b). \tag{2.36}$$

Taking the gradient of (2.36) with respect to x and set to zero we get

$$\begin{aligned} \nabla_x L(x, \nu) = x - \big(z^k - (\nu^k/\rho)\big) + A^T y = 0 \\ \therefore \; A^T y = \big(z^k - (\nu^k/\rho)\big) - x \end{aligned} \tag{2.37}$$

We want to have Ax in order to sub in our constraint $Ax = b$, so we multiply through by A, then solve for y,

$$\begin{aligned} AA^T y = A\big(z^k - (\nu^k/\rho)\big) - Ax = A\big(z^k - (\nu^k/\rho)\big) - b \\ \therefore \; y = (AA^T)^{-1}\big(A\big(z^k - (\nu^k/\rho)\big) - b\big) \end{aligned} \tag{2.38}$$

Substitute y into (2.37) and rearrange for x to get,

$$\begin{aligned} 0 &= x - (z^k - (\nu^k/\rho)) + A^T\Big((AA^T)^{-1}\big(A\big(z^k - (\nu^k/\rho)\big) - b\big)\Big) \\ \therefore\ x &= \big(z^k - (\nu^k/\rho)\big) - A^T\Big((AA^T)^{-1}\big(A\big(z^k - (\nu^k/\rho)\big) - b\big)\Big) \\ &= (I - A^T(AA^T)^{-1}A)(z^k - \nu^k/\rho) + A^T(AA^T)^{-1}b. \end{aligned} \tag{2.39}$$

We have found the x that minimizes the right hand side of the x update in (2.34). Previously in the dual ascent and dual decomposition sections we stopped at a general update expression. This is our first explicit update equation that can be implemented in practice. Notice that some of the operators do not change with iteration and can be precomputed, i.e. AA^T.

The z update in (2.34) is a soft thresholding operator. Starting from a general form of the z update equation (2.34),

$$\underset{a}{\text{minimize}}\ \frac{1}{2}\big\|a-b\big\|_2^2 + c\|a\|_1, \tag{2.40}$$

then taking the gradient with respect to a we get

$$0 \in \underset{a}{\nabla}\left(\frac{1}{2}\big\|a-b\big\|_2^2 + c\|a\|_1\right) = a - b + c\nabla\|a\|_1. \tag{2.41}$$

The ℓ_1 norm is not differentiable at zero but we can consider three cases separately,

$$\begin{array}{lllll} a>0 & \Rightarrow & \nabla\|a\|_1 = 1 & \Rightarrow & a = b - c \\ a=0 & \Rightarrow & \nabla\|a\|_1 = [-c,c] & \Rightarrow & a = b - [-c,c] \\ a<0 & \Rightarrow & \nabla\|a\|_1 = -1 & \Rightarrow & a = b + c, \end{array} \tag{2.42}$$

resulting in a piecewise expression,

$$S_c(b) = \begin{cases} b - c, & b > c \\ 0, & -c \leq b \leq c \\ b + c, & b < -c \end{cases} \tag{2.43}$$

This operator can be implemented, and is visualized in in figure 2.2. Trans-

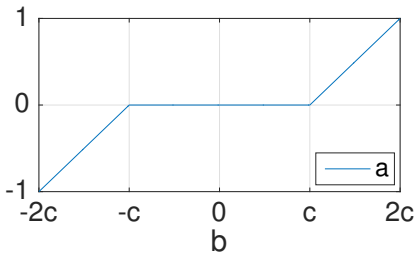

Figure 2.2: Visualization of the soft threshold operator.

lating back to the variables,

$$z^{(k+1)} = \begin{cases} (x^{(k+1)} + \nu^k/\rho) - 1, & (x^{(k+1)} + \nu^k/\rho) > 1 \\ 0, & -1 \leq (x^{(k+1)} + \nu^k/\rho) \leq 1 \\ (x^{(k+1)} + \nu^k/\rho) + 1, & (x^{(k+1)} + \nu^k/\rho) < -1. \end{cases} \tag{2.44}$$

Finally, the dual variable ν update is found by gradient ascent. Considering the gradient of (2.32) with respect to ν, only the third term contributes,

$$\underset{\nu}{\nabla} L_\rho(x, z, \nu) = x - z. \tag{2.45}$$

We can move ν towards the dual maximum by adding the most recent $x-z$ to the previous ν value, as already given in (2.34). We have now found all three updates explicitly and can implement our first iterative distributable algorithm. Figure 2.3 shows simulation results for randomly generated measurement data where the primal and dual residuals (called $rNorm$ and $sNorm$ respectively) converge towards zero ($epsPri$ and $epsDual$ variable

exit conditions which are not necessary for this discussion).

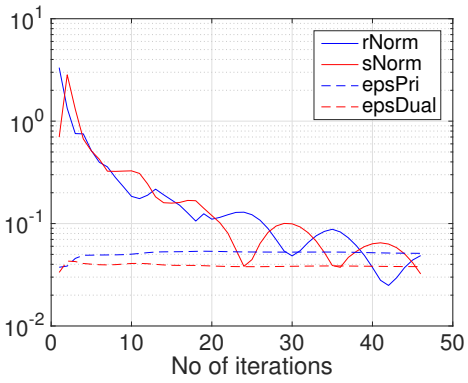

Figure 2.3: The primal and dual residuals $rNorm$ and $sNorm$ respectively converge towards zero as the iteration gets closer to the optimal solution. $epsPri$ and $epsDual$ are the exit conditions for the primal and dual respectively.

We now have an implementable distributed algorithm for solving a convex optimization problem. The limitations of ADMM are still present; the distribution of the primal update can be over a maximum of two nodes, and the dual update can not be distributed. PDMM does not have these limitations. It was recently introduced in 2013 and allows the iterative solution of a convex optimization problem in a fully distributed manner.

2.1.8 PDMM: Primal Dual Method of Multipliers

The primal dual method of multipliers (PDMM, formerly Augmented Bi-Conjugate function, or Bi-ADMM) was introduced by Zhang and Huesdens in 2013 [14] as an alternative to ADMM with the potential for faster convergence and greater flexibility. They aimed to improve ADMM, which solves a constrained min-max or saddle point problem, by formulating an unconstrained and distributable minimization problem that results in an

equivalent solution. The problem statement given in [14] is

$$\begin{aligned} &\text{minimize} && f(x) + g(z) \\ &\text{subject to} && Mx = z. \end{aligned} \tag{2.46}$$

The same authors with the addition of Kleijn study the convergence of PDMM in [15], and generalize the problem statement to

$$\begin{aligned} &\text{minimize} && f(x) + g(z) \\ &\text{subject to} && Ax + Bz = c. \end{aligned} \tag{2.47}$$

We will continue with the more general problem statement (2.47). The primal Lagrangian L_p is given by

$$L_p(x, z, \nu) = f(x) + g(z) + \nu^T(c - Ax - Bz), \tag{2.48}$$

and the dual function by,

$$\begin{aligned} g(\nu) &= \inf_{x,z} L(x, z, \nu) \\ &= \inf_{x,z} f(x) + g(z) + \nu^T(c - Ax - Bz). \end{aligned} \tag{2.49}$$

Taking the gradient with respect to x and z would result in a solution including the gradient operator, so instead we use the convex conjugates f^* and g^* (introduced in section 2.1.1, see [5] for more detail) of f and g to express the two infimums,

$$\begin{aligned} g(\nu) &= \inf_{x,z} \Big\{ f(x) + g(z) + \nu^T(c - Ax - Bz) \Big\} \\ &= \nu^T c + \inf_{x} \Big\{ f(x) - \nu^T Ax \Big\} + \inf_{z} \Big\{ g(z) - \nu^T Bz \Big\} \\ &= \nu^T c - f^*(A^T\nu) - g^*(B^T\nu), \end{aligned} \tag{2.50}$$

where the convex conjugates are given by

$$\begin{aligned} f^*(A^T\nu) &= -\inf_x \ \Big\{f(x) - \nu^T Ax\Big\} \\ g^*(B^T\nu) &= -\inf_z \ \Big\{g(z) - \nu^T Bz\Big\}. \end{aligned} \tag{2.51}$$

It follows immediately that

$$\begin{aligned} f^*(A^T\nu) + f(x) - \nu^T Ax &\geq 0 \\ g^*(B^T\nu) + g(z) - \nu^T Bz &\geq 0, \end{aligned} \tag{2.52}$$

which is commonly referred to as Fenchel's inequality [5]. The dual problem is

$$\sup_\nu \ g(\nu) = \sup_\nu \Big\{\nu^T c - f^*(A^T\nu) - g^*(B^T\nu)\Big\}. \tag{2.53}$$

The conjugate functions are then decoupled by introducing λ via the following optimization,

$$\begin{aligned} &\underset{\lambda}{\text{maximize}} \ \ \lambda^T c - f^*(A^T\nu) - g^*(B^T\lambda) \\ &\text{subject to} \ \ \nu = \lambda, \end{aligned} \tag{2.54}$$

and a second Lagrangian constructed for the dual problem by introducing another multiplier y,

$$L_d(\nu, y) = \lambda^T c - f^*(A^T\nu) - g^*(B^T\lambda) + y^T(\nu - \lambda) \tag{2.55}$$

with $y = Bz$. The primal dual function can now be constructed with $\rho > 0$, and $h_p = \frac{\rho}{2}\|c - Ax - Bz\|_2^2 + \frac{1}{2\rho}\|\nu - \lambda\|_2^2$ enforcing the constraints,

$$\begin{aligned} L_\rho(x, z, \nu, \lambda) &= L_p(x, z, \nu) + L_d(z, \nu, \lambda) + h_p(x, z, \nu, \lambda) \\ &= f(x) + g(z) + \nu^T(c - Ax - Bz) \\ &\quad + \lambda^T c - f^*(A^T\nu) - g^*(B^T\lambda) + (Bz)^T(\nu - \lambda) \\ &\quad + \frac{\rho}{2}\|c - Ax - Bz\|_2^2 + \frac{1}{2\rho}\|\nu - \lambda\|_2^2. \end{aligned} \tag{2.56}$$

Note that as a result of (2.52) and the non-negativity of the two squared norm penalty terms, $L_\rho(x, z, \nu, \lambda) \geq 0$. Minimization is then performed in alternating directions over the variables. The most general updates given in [15] are

$$\begin{aligned} (x^{(k+1)}, \lambda^{(k+1)}) &= \arg\min_x \max_\lambda L_\rho(x, z^{(k)}, \nu^{(k)}, \lambda) \\ (z^{(k+1)}, \nu^{(k+1)}) &= \arg\min_z \max_\nu L_\rho(x^{(k)}, z, \nu, \lambda^{(k)}). \end{aligned} \tag{2.57}$$

The similarity to ADMM is in the alternating direction coordinate descent updates. The difference is the lack of a centralized gradient ascent dual update. This lack of a centralized dual update allows PDMM to be performed in a fully distributed manner. The convergence rate of PDMM is analyzed in [15] and shown to be $\mathcal{O}(K^{-1})$, and to outperform ADMM.

PDMM has another advantage; it converges under an asynchronous update scheme [16]. This is in contrast to ADMM which must maintain the correct update order to guarantee convergence (aynchronous distributed ADMM also addresses these issues but PDMM gives faster convergence). In our distributed beamformer application this reduces the organizational cost as the nodes are no longer restricted to using a global clock.

[15] provided for distributed optimization, but was still limited to two nodes. The next step was generalization of PDMM to arbitrary graphs.

2.1.9 PDMM: Extension to Arbitrary Graphs

The final step in the development of an algorithm for fully distributed beamforming was the extension of PDMM to arbitrary graphs. This was presented by Zhang and Huesdens in [16] in 2015.

To cope with full distribution the model naturally becomes more complicated. Define a network graph $G = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ represent the nodes (vertices), and $\mathcal{E}$ represents the edges between nodes. Each node i has a local closed, proper and convex objective function $f_i(x_i) : \mathbb{R}^{|\mathcal{N}_i|} \to \mathbb{R} \cup \{+\infty\}$, a set of neighboring nodes $\mathcal{N}_i$ that share an edge, and a total number of neighbors including itself given by the cardinality $|\mathcal{N}_i|$. The total number of nodes is given by the cardinality $|\mathcal{V}| = m$.

The separable global objective is defined as

$$\begin{aligned} &\underset{x}{\text{minimize}} \ \sum_{i \in \mathcal{V}} f_i(x_i) \\ &\text{subject to} \ \ A_{i|j} x_i + A_{j|i} x_j = c_{ij}, \ \ \forall (i,j) \in \mathcal{E} \end{aligned} \tag{2.58}$$

where $(c_{ij}, A_{i|j}, A_{j|i}) \in (\mathbb{R}^{n_{ij}}, \mathbb{R}^{n_{ij} \times n_i}, \mathbb{R}^{n_{ij} \times n_j})$ for every edge $(i,j) \in \mathcal{E}$ and are used to enforce the consensus between neighboring nodes. The Lagrangian for the distributed problem is given by,

$$L(x, \lambda) = \sum_{(i,j) \in \mathcal{E}} \lambda_{ij}^T (c_{ij} - A_{i|j} x_i - A_{j|i} x_j) + \sum_{i \in \mathcal{V}} f_i(x_i). \tag{2.59}$$

The Lagrangian is convex in x and concave in λ, where x and λ are the stacked combinations of $x_i, \ \forall i \in \mathcal{V}$ and $\lambda_{ij}, \ \forall (i,j) \in \mathcal{E}$.

As we can see in (2.59), the component of the Lagrangian due to the constraints is directed, that is, there is a Lagrange multiplier from node i to j, and another one from j to i. Let $\lambda_{i|j} \in \mathbb{R}^{n_{ij}}$ represent the multiplier owned by node i relating to node j. Let λ_i represent the vertical concatenation of these $\lambda_{i|j}, \forall j \in \mathcal{N}_i$. Let A_i represent the vertical concatenation of $A_{i|j}, \forall j \in \mathcal{N}_i$. Then, following the construction of (2.56), the augmented

primal dual function is constructed for an arbitrary network graph,

$$L(x,\lambda) = \sum_{i\in\mathcal{V}} \Big[f_i(x_i) - \sum_{j\in\mathcal{N}_i} \lambda_{j|i}^T \big(A_{i|j}x_i - c_{ij}\big) - f_i^*(A_i^T\lambda_i)\Big] + \sum_{(i,j)\in\mathcal{E}} \left[\frac{1}{2}\big\|A_{i|j}x_i + A_{j|i}x_j - c_{ij}\big\|_2^2 - \frac{1}{2}\big\|\lambda_{i|j} - \lambda_{j|i}\big\|_2^2\right], \tag{2.60}$$

and again following [15] we get the updates,

$$(x^{(k+1)}, \lambda^{(k+1)}) = \arg\min_x \max_\lambda L(x,\lambda). \tag{2.61}$$

Substituting in $L(x,\lambda)$, the updates are given by,

$$x_i^{(k+1)} = \arg\min_{x_i} \Bigg[\sum_{j\in\mathcal{N}_i} \frac{1}{2}\|A_{i|j}x_i + A_{j|i}x_j^{(k)} - c_{ij}\|_2^2 - x_i^T\Bigg(\sum_{j\in\mathcal{N}_i} A_{i|j}^T\lambda_{j|i}^{(k)}\Bigg) + f_i(x_i)\Bigg], \quad \forall i\in\mathcal{V} \tag{2.62}$$

$$\lambda_i^{(k+1)} = \arg\min_{\lambda_i} \Bigg[\sum_{j\in\mathcal{N}_i} \Bigg(\frac{1}{2}\|\lambda_{i|j} - \lambda_{j|i}^{(k)}\|_2^2 + \lambda_{i|j}^T A_{j|i}x_j^{(k)} - \lambda_{i|j}^T c_{ij}\Bigg) + f_i^*(A_i^T\lambda_i)\Bigg], \quad \forall i\in\mathcal{V}. \tag{2.63}$$

These updates are for synchronous updating where all nodes update at each iteration. It is possible to run this updating scheme as an asynchronous fashion where a single node updates in each iteration. If some assumptions are met, i.e. that all nodes update and the frequency is similar for all nodes, then PDMM will still converge [16].

Zhang, Heusdens and Kleijn provide a convergence analysis in [16] for both the synchronous and asynchronous case, and demonstrate the algorithm with an application to distributed averaging. They show that both synchronous and asynchronous schemes have a convergence rate of $\mathcal{O}(K^{-1})$ (where K is the iteration number) for properly defined objective functions. PDMM is shown to converge faster than ADMM in the dis-

tributed averaging case.

[16] also provides a simplified dual update for the case where $P_{p,ij} = P_{p,ij}^{-1}$. P is used to define a generalized ℓ_2 norm. In our application $P = I$ and therefore meets the condition so the simplified update can be used,

$$\lambda_{j|i}^{(k+1)} = \lambda_{j|i}^{(k)} + c_{ij} - A_{j|i}x_j^{(k)} - A_{i|j}x_i^{(k+1)}. \tag{2.64}$$

We now have a fully distributed algorithm capable of optimizing a constrained convex separable objective. This algorithm enables optimal distributed beamforming, which will be discussed in section 2.2.8. The background of distributed convex optimization over arbitrary graphs has been given, we now turn to our application; acoustic beamforming. The following sections give background on beamforming and then draw distributed convex optimization and acoustic beamforming together.

2.2 Beamforming for Acoustics

Here we provide details necessary for the implementation of distributed acoustic beamformers in the context of our proposed distributed BFOP calculation. The STFT, conjugate symmetry, conversion from real to complex domain, and complex gradient are discussed. This is followed by the development of beamforming from delay and sum, to optimal MVDR, adaptive beamforming following Frost, and finally arriving at sparse distributed MVDR using PDMM.

2.2.1 Time Frequency Domain

Broadband beamformers take the structure of a finite impulse response filter which can be computed in the frequency domain. Transformation to the frequency domain is achieved using a DFT. The DFT requires the complete signal to be available but this is not possible in real time audio

processing as new signal arrives continuously. We would also like to allow for a dynamic scenario where the target and interference sources are allowed to move, so require a dynamic beamformer that can reoptimize the weights at some time interval. The STFT breaks the incoming audio into windowed sections and takes the DFT of each section. The result is one frequency spectrum per windowed section, called the time frequency spectrum [39].

The simplest window is the rectangular window. This window function distorts the spectrum of the windowed audio. The rectangle window becomes a sinc function in the frequency domain. If we pass a sine wave through a rectangular window and take the DFT we will find the spectrum distorted by the sinc, as shown in figure 2.4. Choosing a window requires trading between sidelobe attenuation, and main lobe width. The window throughout this thesis is the Hann window (sometimes called Hanning, not to be confused with Hamming). The Hann window is a cosine window with sidelobes that taper off quickly and can be made to sum to one with a 50% overlap. The summation behaviour of the Hann window is shown in figure 2.5. The Hann window is defined by,

$$w(n) = 0.5 - 0.5\cos\left(\frac{2\pi n}{N-1}\right), \tag{2.65}$$

where n is the sample number and N is the window length in samples. Error in the reconstructed signal can be reduced by windowing in both the analysis and synthesis operations [40]. The combined windowing operations must still sum to one for a unitary transform so the square root of the Hann window is then used for both analysis and synthesis.

Following sections discuss phase delaying the frequency domain signal. The window length must be chosen to be longer than any of the required delays to avoid distortion in the synthesized signal.

It is important to get the alignment of consecutive windows correct. Observing the window function (2.65) we see that the window starts at

0 when $n = 0$, and arrives again at 0 when $n = N - 1$. The window maximum is 1 and occurs when the argument of the cosine is exactly π, that is, when $n = (N-1)/2$. n is a sample index so $n \in \mathbb{Z}_+$, therefore N must be odd. The correct alignment of the windows can be achieved by placing the start of each window at

$$n = k\left(\frac{N-1}{2}\right) + 1, \quad k \in [0, 1, 2, \dots]. \tag{2.66}$$

Incorrect window delays can result in unexpected results. Note that here we have considered n starting from 0. In some languages, i.e. Matlab, the indexing starts at 1 and this will have to be taken into account.

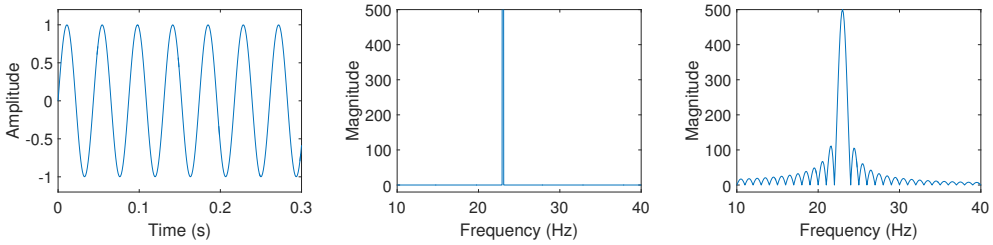

Figure 2.4: (a) Original signal: a 23 Hz sine wave. (b) DFT of the sine wave, prior to windowing. The DFT is a clean and accurate spectrum of the original signal. Note that the magnitude axis has been restricted for comparison with (c) is larger than what is visible. (c) DFT of the sine wave post rectangle windowing. The spectrum has been distorted by windowing.

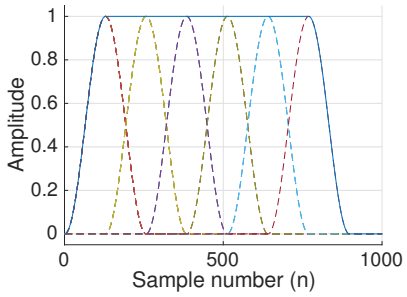

Figure 2.5: The Hann window sums to 1 provided the correct delays are used.

The incoming time domain signal is multiplied with the shifted window function and passed through a DFT. Let $x[n]$ represent the observation signal with n indexing the sample. The typical DFT analysis/synthesis pair is given by

$$\begin{aligned} X[k] &= \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N}, \quad k \in \mathbb{Z} \\ x[n] &= \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j2\pi kn/N}, \end{aligned} \tag{2.67}$$

where the $1/N$ term conserves signal energy. Note that there are other possible scalings which also preserve signal energy (see appendix A.2). Including the delayed window gives the time frequency spectrum,

$$X[k, m] = \sum_{n=0}^{N-1} w[n - Lm] x[n] e^{-j2\pi kn/N}, \quad k \in \mathbb{Z} \tag{2.68}$$

where L is the number of samples to shift each consecutive window, and $m \in [0, 1, \ldots, M-1]$ is a positive integer that indexes each window. Synthesis of the original signal from the time frequency spectrum requires an overlap add operation [41]. An inverse DFT is taken of each section, i.e. for each value of m. Each section is again multiplied with the square root Hann window and overlapped with correct time alignment, the sections are then summed resulting in the original signal.

This process enables time varying frequency domain filtering of continuously arriving audio signals. Also, by using Cooley and Tukey's 1965 Fast Fourier Transform (FFT) algorithm [42] this filtering can be achieved efficiently.

The summation of the windows in figure 2.5 shows that the summation to one is only true from the midpoint of the first window to the midpoint of the last window. That is, windowing distorts the start and end of the signal. This can be addressed by zero padding the beginning and end of

the observation signal such that the observed signal does not fall within the first or last half window. The first and last half windows then act only on zeros.

Once the observations are in the time frequency domain, we aim to delay the observations in order to phase align the target signal. The time delays from target signal to sensor exist in a continuous domain, while the sampled audio exists in a discrete domain. It is generally not possible to phase align the target signal using integer valued sample delays. Non-integer delays can be achieved using frequency domain phase shifts. In order to do so, conjugate symmetry can be used to ensure that the phase shifted time domain signal is guaranteed real.

2.2.2 Conjugate Symmetry

Our aim is to phase align the sensor observations of the target signal using phase shifts. Implementing frequency domain phase shifts can cause the shifted time domain signal to become complex. The conjugate symmetry property of the Fourier transform of a real signal can be used to make assumptions about the spectrum which allow us to ensure that the post phase shift time domain signal is real.

Let $x[n] \in \mathbb{R}^N$, the DFT of $x[n]$ is $X[k] \in \mathbb{C}^N$, and $(.)^*$ indicates the complex conjugate. Conjugate symmetry is defined by

$$X[N-k] = X^*[k]. \tag{2.69}$$

We can derive this property for real x by starting from the left hand side

of (2.69) and arriving at the right hand side,

$$\begin{aligned} X[N-k] &= \sum_{n=0}^{N-1} x[n] e^{-j2\pi(N-k)n/N} \\ &= \sum_{n=0}^{N-1} x[n] e^{-j2\pi n} e^{j2\pi kn/N} \\ &= \sum_{n=0}^{N-1} \left(x^*[n] e^{-j2\pi kn/N} \right)^* \\ &= \left(\sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N} \right)^* \\ &= X^*[k], \end{aligned} \tag{2.70}$$

where we have used that x is real $x[n] = x^*[n]$, $e^{-j2\pi n} = 1$ for $n \in \mathbb{Z}$, and that the sum of conjugates is equal to the conjugate of the sum. To see the last property, let $(a = a_r + ja_i, b = b_r + jb_i) \in \mathbb{C}$ where $j^2 = -1$.

$$\begin{aligned} (a+b)^* &= \big((a_r + ja_i) + (b_r + jb_i)\big)^* \\ &= \big((a_r + b_r) + j(a_i + b_i)\big)^* \\ &= (a_r + b_r) - j(a_i + b_i) \\ &= (a_r - ja_i) + (b_r - jb_i) \\ &= a^* + b^*. \end{aligned} \tag{2.71}$$

Audio beamforming starts with a real signal (sampled time domain audio can only be real valued), which therefore has conjugate symmetry in the frequency domain coefficients. This allows us to throw away half of the coefficients, as they can be perfectly synthesized from the other half. The DC component can also be removed, and we assume that the Nyquist sampling filter guaranteed no energy at half of the sampling frequency $f_s/2$ so the $\pm f_s/2$ components of the spectrum can be truncated. We can now phase shift the truncated spectrum, recreate the missing spectrum

using conjugate symmetry and values of 0 at DC and $\pm f_s/2$, then take the inverse transform and arrive at a real time domain signal.

Another issue which arises in practice is the computation of the beamformer in the complex domain. Some sources within the literature, for example [34], calculate the beamformer in the real domain, with an assumed conversion from complex to real domains. Conversion between the two is described next.

2.2.3 Conversion from Complex to Real Domain

In order to follow the distributed beamforming literature, it is necessary to be able to convert our complex domain problem to the real domain. Let $a \in \mathbb{C}^N$ with the standard inner product

$$a^H a = \sum_{k=1}^{N} a_k^* a_k = \sum_{k=1}^{N} (x_k - jy_k)(x_k + jy_k) = \sum_{k=1}^{N} x_k^2 + y_k^2, \tag{2.72}$$

where x_k, y_k are the real and imaginary components of a_k i.e. $a = x + jy$, j is the imaginary unit i.e. $j^2 = -1$, $(.)^*$ is the complex conjugate operator, and $(.)^H$ is the hermitian operator. If we transform a into a vector $\hat{a} \in \mathbb{R}^{2N}$ using

$$\hat{a} = \begin{bmatrix} x \\ y \end{bmatrix}, \tag{2.73}$$

then we can calculate an equivalent inner product

$$\hat{a}^T \hat{a} = \begin{bmatrix} x^T & y^T \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = x^T x + y^T y = \sum_{k=1}^{N} x_k^2 + y_k^2 = a^H a. \tag{2.74}$$

Note that this does not extend to inner products between two vectors that result in a complex output. For example, assume $b = v + jw \in \mathbb{C}^N$ and

$\hat{b} = [v^T \; w^T]^T$ then

$$a^H b = \sum_{k=1}^{N} xv + yw + j(xw - yv) \neq \hat{a}^T \hat{b}. \tag{2.75}$$

Conversion is also possible for a complex linear mapping. Let $D = (D_r + jD_i) \in \mathbb{C}^{N \times N}$ and

$$\hat{D} = \begin{bmatrix} D_r & -D_i \\ D_i & D_r \end{bmatrix} \in \mathbb{R}^{2N \times 2N}. \tag{2.76}$$

Then

$$\begin{aligned} \hat{D}\hat{a} &= \begin{bmatrix} D_r & -D_i \\ D_i & D_r \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \\ &= \begin{bmatrix} D_r x - D_i y \\ D_i x + D_r y \end{bmatrix} \\ &= \begin{bmatrix} (Da)_r \\ (Da)_i \end{bmatrix}. \end{aligned} \tag{2.77}$$

Da can then be recovered by

$$Da = \begin{bmatrix} 1 & j \end{bmatrix} \hat{D}\hat{a} = (Da)_r + j(Da)_i = (D_r x - D_i y) + j(D_i x + D_r y). \tag{2.78}$$

Having established that conversion from the complex domain to the real domain is possible in the distributed beamforming context, we now have the option to consider derivations and algorithms entirely in the real domain.

2.2.4 Complex Gradient and Stationary Points

For beamforming we will aim to minimize an objective that will be a real valued function of a complex variable. Finding the minimum of a function is usually achieved by taking the derivative and setting it to zero, but

when we have a complex variable the function is no longer analytic and the derivative does not exist [43]. The derivative can be found by treating the function as having two independent variables, i.e. x and x^*. See [43, 44, 45] for further reference. The gradient is given by,

$$\nabla f(x) := 2\frac{\delta f(x, x^*)}{\delta x^*} \tag{2.79}$$

which we can apply to a complex quadratic,

$$\begin{aligned} f(x) &= x^H A^H A x \\ \nabla f(x) &= 2A^H A x. \end{aligned} \tag{2.80}$$

The term $A^H A$ has been used here to highlight the structure of the operator, which has a real diagonal and is conjugate symmetric. In our application, this operator will be a covariance matrix that will satisfy these structural requirements and therefore allow an analytic gradient to be found. However, other elements of our application such as complex inner products and complex operators do not fit the criteria. In order to cope with these elements we can transform them from $\mathbb{C}^N$ to $\mathbb{R}^{2N}$ and treat the real and complex parts as independent variables as discussed in section 2.2.3.

2.2.5 Delay and Sum

Having discussed the necessary auxiliary tools for the implementation of frequency domain beamforming, we turn to the background of beamforming itself. The story begins with the delay and sum beamformer [46].

A talker's voice travels through air at a constant velocity. If two microphones are placed at different distances from the talker, the speech will arrive at the near microphone first, and the more distant microphone second. It seems intuitive that we could sum the two microphones to increase the SNR of the target signal. However, the different distances between the talker and the two microphones means that simply summing the signals

will make little difference as they will be out of phase and effectively independent. The delay and sum beamformer adds a specific delay to each microphone to exactly offset the delay caused by the different distances. The delayed signals are then summed. Because the signals are now phase aligned, they sum constructively and improve the SNR of the estimated target speech. Figure 2.6 gives a visual representation of the system, illustrating that the target signal is constructively summed, while the interference signal is not.

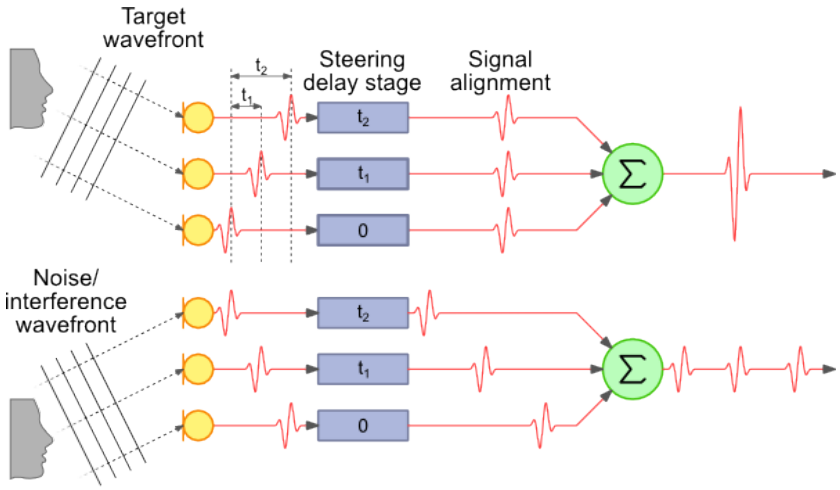

Figure 2.6: The delay and sum beamformer introduces delays that phase align the desired target signal, then sums the delayed observations. The alignment causes the target signal to sum constructively while (almost) all other sources are not summed constructively. (Image credit http://www.labbookpages.co.uk/audio/beamforming/delaySum.html)

The delay and sum beamformer is effective, but not optimal in the presence of interference. Capon introduced the minimum variance distortionless response (MVDR) beamformer in [21], which uses optimal filter weights to place a null at the location of each point source interferer (the number of point sources is limited by the degrees of freedom in the system). This results in improved performance over the delay and sum beamformer. The following section describes Capon's beamformer.

2.2.6 Optimal MVDR

MVDR beamformers aim to minimize the output variance while requiring unity gain across all frequencies in the look direction. We assume the observations are independent across window and frequency indices and can therefore omit them for clarity.

Let $Y \in \mathbb{C}$ be a random variable representing the frequency domain output of the beamformer, which is a weighted sum of observations, $Y = w^H X$. The weights $w \in \mathbb{C}^M$, and the observations are a random variable $X \in \mathbb{C}^M$, where M is the total number of sensors. Each observation is a sum of the attenuated and phase shifted target signal plus noise and interference $X = dS + N$. $d \in \mathbb{C}^M$ is the look direction vector representing the phase shift and attenuation, $S \in \mathbb{C}$ is a random variable representing the target source coefficient, and $N \in \mathbb{C}^M$ is a random variable representing the noise and interference at each sensor node. The noise is assumed to be a combination of attenuated and phase shifted point source interference signals (i.e. other talkers) and independent noise at each node (perhaps thermal noise in the sensor circuitry). The output variance is given by

$$\begin{aligned}
\mathrm{E}[Y^H Y] &= \mathrm{E}[(w^H X)^H (w^H X)] \\
&= \mathrm{E}[(w^H X)(w^H X)^H] \\
&= w^H \mathrm{E}[X X^H] w \\
&= w^H \mathrm{E}[(dS + N)(dS + N)^H] w \\
&= w^H \big(\mathrm{E}[S^2] dd^H + \mathrm{E}[N N^H]\big) w \\
&= \mathrm{E}[S^2] w^H dd^H w + w^H R w
\end{aligned} \tag{2.81}$$

where $R = \mathrm{E}[NN^H]$ is the power spectral density matrix of the unwanted noise and interference signals. The power spectral density is the frequency domain equivalent of the spatial covariance. We will refer to R as a covariance matrix from here.

MVDR beamformers aim to minimize the output variance while con-

straining the gain to unity in the look direction across all frequency bins. The first term of the output variance expressed in (2.81) is not dependent on w due to the unity gain look direction constraint, so we only need to minimize the second term. The optimization problem is

$$\begin{aligned} &\text{minimize} \;\; \frac{1}{2} w^H R w \\ &\text{subject to} \;\; w^H d = 1. \end{aligned} \tag{2.82}$$

The method of Lagrangian multipliers can be used to convert (2.82) from a constrained form to an unconstrained form,

$$L(w, w^H, \nu) = \frac{1}{2} w^H R w - \nu(w^H d - 1) - \nu^*(d^H w - 1), \tag{2.83}$$

where $(.)^*$ represents the complex conjugate operator. Taking the complex gradient (see section 2.2.4) with respect to w^H and equating to 0 we get

$$\begin{aligned} \underset{w^H}{\nabla} L(w, w^H, \nu) &= Rw - \nu d = 0 \\ \therefore \;\; w &= \nu R^{-1} d. \end{aligned} \tag{2.84}$$

The weights have to meet the constraint, so substituting (2.84) into (2.82),

$$\begin{aligned} d^H w = \nu d^H R^{-1} d &= 1 \\ \therefore \;\; \nu &= \frac{1}{d^H R^{-1} d} \\ \therefore \;\; w &= \frac{R^{-1} d}{d^H R^{-1} d}. \end{aligned} \tag{2.85}$$

Equation (2.85) is the well known optimal MVDR beamformer weights. The true noise covariance has been used to derive the optimal weights, in practice the true noise covariance is unavailable. An unbiased estimate of the observation covariance can be made by averaging observation covariances, this is sometimes referred to as a sample matrix inversion (SMI)

beamformer [47]. Matrix inversion can also be problematic, and is commonly addressed by introducing additional uncorrelated noise. If the scenario is dynamic we need to recalculate the optimal weights regularly. It is common to use an adaptive algorithm to maintain the optimal weights rather than the analytic optimal weights given in (2.85). Frost introduced an adaptive algorithm for implementation of an minimum variance beamformer in [24], which we discuss in the following section.

2.2.7 Frost's Adaptive Algorithm

Frost's 1972 paper [24] describes an adaptive algorithm for implementing an LCMV (or MVDR) broadband beamformer. LCMV is a generalization of MVDR that can have any set of linear constraints. Frost also gave a geometric interpretation of the algorithm which gives valuable insight into the algorithm and sources of error, in particular accumulative error. The advantages of Frost's algorithm were that it required no prior knowledge of the signal second order statistics and did not accumulate errors in the weight vector, while maintaining a hard constraint.

The LCMV optimization problem is the same as for MVDR but with the unity gain constraint replaced with any constant f,

$$\begin{aligned} &\text{minimize } \frac{1}{2} w^H R w \\ &\text{subject to } w^H d = f. \end{aligned} \tag{2.86}$$

The optimal weights are derived using the same method outlined in the previous section to get

$$w = \frac{f R^{-1} d}{d^H R^{-1} d}, \tag{2.87}$$

and the optimal BFOP is given by,

$$Y = w^T X = \left(\frac{f R^{-1} d}{d^T R^{-1} d} \right)^T X. \tag{2.88}$$

The true R is not known, so the analytic equation for optimal w cannot be used. Instead we iteratively approach the minimum using gradient descent and an approximation of the noise covariance,

$$w^{(k+1)} := w^{(k)} - \mu \underset{w}{\nabla} L(w^{(k)}, \nu) = w^{(k)} - \mu(Rw^{(k)} - \nu d), \tag{2.89}$$

where μ controls the step size. The constraint still has to be met so

$$\begin{aligned} f &= d^T w^{(k+1)} \\ &= d^T \big(w^{(k)} - \mu(Rw^{(k)} - \nu d) \big) \\ \therefore \ \nu &= \frac{1}{\mu d^T d} \big(f - d^T w^{(k)} + \mu d^T R w^{(k)} \big) \end{aligned} \tag{2.90}$$

Subbing ν back into (2.89),

$$\begin{aligned} w^{(k+1)} &= w^{(k)} - \mu \left(Rw^{(k)} - \left(\frac{1}{\mu d^T d} \big(f - d^T w^{(k)} + \mu d^T R w^{(k)} \big) \right) d \right) \\ &= P(\mu R - I) w^{(k)} + F, \end{aligned} \tag{2.91}$$

where I is the identity matrix, P and F are precomputable and given by

$$P = \frac{1}{d^T d} d d^T - I, \qquad F = \frac{f}{d^T d} d. \tag{2.92}$$

As we do not have the actual noise covariance, Frost suggests approximating $R^{(k)}$ with the covariance of the observations, that is $R^{(k)} = x^{(k)} x^{(k)T}$. In practice it is common to use a running average over some number of iterations. Here we have used x rather than X to indicate that the observation is no longer a random variable, but an observation. Subbing $R^{(k)} = x^{(k)} x^{(k)T}$

into (2.91) we get,

$$\begin{aligned} w^{(k+1)} &= P(\mu R^{(k)} w^{(k)} - w^{(k)}) + F \\ &= P(\mu x^{(k)} x^{(k)T} w^{(k)} - w^{(k)}) + F \\ &= P(\mu x^{(k)} y^{(k)} - w^{(k)}) + F. \end{aligned} \tag{2.93}$$

The update equation (2.93) uses available information only and is implementable. Figure 2.7 shows simulated gain curves for each frequency bin as a function of direction for a beamforming problem solved iteratively using (2.93). The spatial filtering achieved by the adapted weights vector has left the target unattenuated in all frequency bins, while heavily attenuating the interferer in all frequency bins. The different wavelengths of each frequency bin causes the spatial response in each bin to be different, with the exception of the look direction which is constrained to unity across all frequency bins.

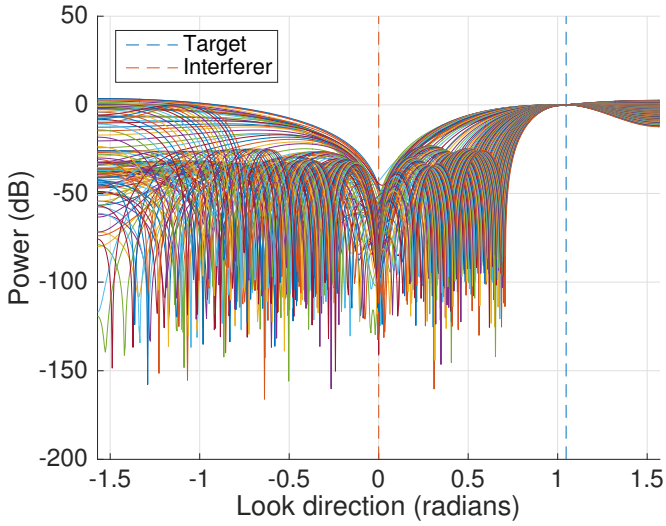

Figure 2.7: Frost's adaptive algorithm was used to find the optimal LCMV weights. The spatial response of the beamformer is shown for a linear sensor array. The target direction has $0\,\mathrm{dB}$ gain across all frequency bins, while the interference direction is heavily attenuated for all frequency bins. Note that each curve represents the spatial response of an individual frequency bin.

Having implemented an adaptive optimal frequency domain beamformer, we have covered the background leading up to distributed beamforming. The following section describes sparse distributed MVDR beamforming using PDMM, as introduced by [34].

2.2.8 Sparse Distributed MVDR Using PDMM

O'Connor, Kleijn and Abhayapala presented an algorithm for applying PDMM to the distributed iterative calculation of a sparse MVDR beamformer in [34]. An ℓ_1 norm regularization term is used to encourage small beamformer weights to zero, noting that the zero weights tend to belong to nodes far from the target source. The optimization effectively chooses a subnetwork of nodes that are close to the target source, though all nodes are still involved in the optimization.

The objective function is derived following [28] using the logarithm of the maximum a posteriori (MAP) estimation of the weights vector, with the addition of ℓ_1 regularization via virtual nodes. Let $w \in \mathbb{R}^M$ be the beamformer weights, $R \in \mathbb{R}^{M\times M}$ is the observation covariance matrix, $d \in \mathbb{R}^M$ the look direction vector, and $\alpha \in \mathbb{R}$ is a scalar. The minimization of the global MVDR objective with an ℓ_1 regularization term is given by,

$$\text{minimize } \frac{1}{2}w^T Rw - d^T w + \alpha\|w\|_1. \tag{2.94}$$

In order to distribute this objective, nodes more than two hops apart are assumed uncorrelated (discussed in section 3.3) allowing the decomposition of the first two terms into a distributed consensus form,

$$\begin{aligned} \text{minimize } & \sum_{k\in\mathcal{N}} \frac{1}{2}w_k^T R_k w_k - d_k^T w_k \\ \text{subject to } & A_{k|l}w_k + A_{l|k}w_l = 0,\ \forall\ (k,l)\ \in \mathcal{E} \end{aligned} \tag{2.95}$$

where $w_k \in \mathbb{R}^{|\mathcal{N}_k|}$ is the local beamformer weights at node k, $\mathcal{N}_k$ is node

k's neighborhood, $R_k \in \mathbb{R}^{|\mathcal{N}_k| \times |\mathcal{N}_k|}$ is the local covariance matrix at node k, $d_k \in \mathbb{R}^{|\mathcal{N}_k|}$ is a local look direction vector that is all zeros except for the element that relates to node k's look direction, $A_{k|l} \in \mathbb{R}^{2 \times |\mathcal{N}_k|}$ and $A_{l|k} \in \mathbb{R}^{2 \times |\mathcal{N}_l|}$ contain $1, -1, 0$ and are designed to ensure consensus between neighboring nodes.

As $\|w\|_1 = \sum_k |w_k|$, the regularization term can be distributed by each node taking the absolute value of its weight estimate relating to itself. A set of virtual nodes $\mathcal{V}$ where $|\mathcal{V}| = |\mathcal{N}|$ are introduced to incorporate the regularization, as the minimization of the ℓ_1 norm must be treated differently to the minimization of the quadratic. Each node has the local optimization

$$\begin{aligned} &\text{minimize} \quad \sum_{k=1}^{|\mathcal{N}|+|\mathcal{V}|} f_k(x_k) \\ &\text{subject to} \quad A_{k|l} w_k + A_{l|k} w_l = 0, \ \forall \ (k,l) \ \in \mathcal{E} \end{aligned} \tag{2.96}$$

where

$$f_k(x_k) = \begin{cases} \frac{1}{2} x_k^T R_k x_k - d_k^T x_k, & \text{for } k \in \mathcal{N} \\ \alpha |x_k|, & \text{for } k \in \mathcal{V}. \end{cases} \tag{2.97}$$

The objective and constraints are used to create the augmented primal dual function as per [16]. Minimizing over the primal variable and maximizing over the dual variable for both the real and virtual nodes results in the update equations where i indexes the iteration,

$$w_k^{(i+1)} := \begin{cases} (\sum_{l \in \mathcal{N}_k} A_{k|l}^T A_{k|l} + R_k)^{-1} \\ \quad (\sum_{l \in \mathcal{N}_k} A_{k|l}^T (\lambda_{l|k}^{(i)} - A_{l|k} x_l^{(i)}) + d_k), & \text{for } k \in \mathcal{N} \\ (-b_k + \text{sign}(b_k) \text{min}(|b_k|, \alpha))(A_{k|l}^T A_{k|l})^{-1}, & \text{for } k \in \mathcal{V} \end{cases} \tag{2.98}$$

$$\lambda_{k|l}^{(i+1)} := \lambda_{l|k}^{(i)} - A_{k|l} x_k^{(i+1)} - A_{l|k} x_l^{(i)}, \qquad \text{for } k \in (\mathcal{N}, \mathcal{V}), l \in \mathcal{N}_k, \tag{2.99}$$

where

$$b_k = 2w_l^{(i)T} A_{l|k}^T A_{k|l} - A_{k|l}^T \lambda_{l|k}^{(i)}. \tag{2.100}$$

Each node calculates updates either asynchronously or synchronously by communicating locally with its neighbors and the global BFFWs converge towards the MVDR optimal weights. The global BFFWs are the concatenation of each node's estimate of its own BFFW. This weight vector can then be used to calculate the BFOP by each node sending its weighted observation $w_k x_k$ to a central location for global computation of the BFOP, or alternatively by using the distributed BFOP calculation proposed in chapter 4.

This chapter presented the background necessary for the implementation of distributed convex optimization, acoustic beamforming, and our proposed distributed BFOP calculation. The following chapter gives some auxiliary results which are useful in the context of distributed acoustic beamforming but which do not belong with the primary result. Chapter 4 then presents our proposed distributed BFOP calculation with analysis and experimental results.

Chapter 3

Auxiliary Results

This section contains some auxiliary results that are useful in the context of the thesis but do not directly contribute to the primary result presented in chapter 4. A proof is given for the preservation of positive semidefiniteness in covariance conversion from the complex domain to the real domain of twice the dimension as discussed in section 2.2.3. Theoretical covariance based on the free space model is given, with discussion and some alternatives. Ensuring that the distributed and centralized problems are equivalent is essential in distributed problems, an example of forcing equivalence is given. The trade off between observing a signal from near and far on a sensor array is considered and raises the possibility of an optimal distribution of sensor nodes. Diffuse noise coherence and the implications for acoustic sensor arrays is considered. Finally, a relationship is derived between the number of nodes in consecutive layers which does not depend on the distance from the centre or the node density.

3.1 Covariance Conversion From $\mathbb{C}^{N\times N}$ to $\mathbb{R}^{2N\times 2N}$ Preserves Positive Semidefiniteness

The literature sometimes assumes a common mapping from the complex domain to the real domain of twice the dimension, as discussed in section 2.2.3. In the complex domain the MVDR objective is expressed as a quadratic i.e. $w^H Rw$ where $w \in \mathbb{C}^N$ is the variable and $R \in \mathbb{C}^{N\times N}$ is the covariance, and convex optimization is used to minimize the objective. Convex optimization is only applicable if the objective function is convex, and for that to be true R must be positive semidefinite. The original complex R is guaranteed to be positive semidefinite precisely because it is a covariance matrix, but does it remain positive semidefinite after mapping to the real domain of twice the dimension? In this section we prove that the converted covariance matrix on $\mathbb{R}^{2N\times 2N}$ is also positive semidefinite.

Let $R = rr^H = Q\Lambda Q^H \in \mathbb{C}^{N\times N}$, where the columns of Q contain the eigenvectors of R, and Λ is a diagonal matrix containing the eigenvalues of R. As R is hermitian, the elements of Λ are real (see appendix A.1 for proof). Then

$$\begin{aligned} R &= Q\Lambda Q^H \\ &= \sum_{i=1}^{N} \lambda_i q_i q_i^H \\ &= \sum_{i=1}^{N} \lambda_i (x_i + jy_i)(x_i^T - jy_i^T) \\ &= \sum_{i=1}^{N} \lambda_i \Big((x_i x_i^T + y_i y_i^T) + j(y_i x_i^T - x_i y_i^T) \Big) \end{aligned} \tag{3.1}$$

where $\lambda_i = \Lambda_{i,i}$, $q_i \in \mathbb{C}^N$ is the ith column vector of Q, $q_i = x_i + jy_i$, and

$j^2 = -1$. The real and imaginary parts of R are

$$R_r = \sum_{i=1}^{N} \lambda_i (x_i x_i^T + y_i y_i^T)$$
$$R_i = \sum_{i=1}^{N} \lambda_i (y_i x_i^T - x_i y_i^T). \tag{3.2}$$

Let $\hat{R} \in \mathbb{R}^{2N \times 2N}$ be the converted covariance (based on section 2.2.3), then

$$\begin{aligned}
\hat{R} &= \begin{bmatrix} \sum_{i=1}^{N} \lambda_i (x_i x_i^T + y_i y_i^T), & -\sum_{i=1}^{N} \lambda_i (y_i x_i^T - x_i y_i^T) \\ \sum_{i=1}^{N} \lambda_i (y_i x_i^T - x_i y_i^T), & \sum_{i=1}^{N} \lambda_i (x_i x_i^T + y_i y_i^T) \end{bmatrix} \\
&= \sum_{i=1}^{N} \lambda_i \begin{bmatrix} (x_i x_i^T + y_i y_i^T), & -(y_i x_i^T - x_i y_i^T) \\ (y_i x_i^T - x_i y_i^T), & (x_i x_i^T + y_i y_i^T) \end{bmatrix} \\
&= \sum_{i=1}^{N} \lambda_i \left(\begin{bmatrix} (x_i x_i^T), & (-y_i x_i^T) \\ (y_i x_i^T), & (x_i x_i^T) \end{bmatrix} + \begin{bmatrix} (y_i y_i^T), & (x_i y_i^T) \\ (-x_i y_i^T), & (y_i y_i^T) \end{bmatrix} \right) \\
&= \sum_{i=1}^{N} \lambda_i \left(\begin{bmatrix} x_i \\ y_i \end{bmatrix} \begin{bmatrix} x_i^T & y_i^T \end{bmatrix} + \begin{bmatrix} y_i \\ -x_i \end{bmatrix} \begin{bmatrix} y_i^T & -x_i^T \end{bmatrix} \right).
\end{aligned} \tag{3.3}$$

As $\hat{R}$ can be expressed as the sum of matrices of form gg^T, which are always positive semidefinite, $\hat{R}$ is positive semidefinite. Note that gg^T is always positive semidefinite as $h^T g g^T h = (h^T g)^2 \geq 0$.

3.2 Theoretical Covariance Based on Distance

There are a number of approaches to finding the sensor network covariance matrix, such as the true noise covariance, the true observation covariance, the sampled observation covariance, and randomly generated covariance. Here we derive the theoretical covariance matrix based only on the distance between the sensors and the target.

Consider the setup shown in figure 3.1. The target signal $t \in \mathbb{C}$ propagates through free space to sensors 1 and 2. The target is is located at

spatial coordinates $l_t \in \mathbb{R}^3$, and the sensors at $l_1, l_2 \in \mathbb{R}^3$, so the distance from t to sensor i is $\|l_t - l_i\|_2$.

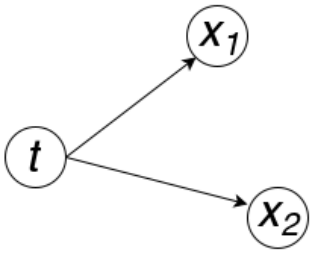

Figure 3.1: Target signal t and two sensor nodes.

Let the target signal $t = e^{-j2\pi f\tau}$, with $j^2 = -1$, f Hz is the frequency, and τ is time in seconds. The target arrives at each sensor attenuated and with a phase shift from the free space transfer function h,

$$x_i = h_i t, \qquad h_i = \frac{e^{-j\phi_i}}{4\pi\|l_t - l_i\|_2}. \tag{3.4}$$

The phase shift ϕ_i is the number of wavelengths in the distance between target t the sensor i,

$$\phi_i = \frac{\|l_t - l_i\|_2}{\lambda} = \frac{\|l_t - l_i\|_2 f}{c}, \tag{3.5}$$

where $\lambda = c/f$ is the wavelength in metres and $c\ \mathrm{m\,s^{-1}}$ is the speed of sound. The spatial covariance R_S with entries r_{ik} is

$$\begin{aligned} r_{ik} &= x_i x_k^* \\ &= (h_i t)(h_k t)^* \\ &= h_i h_k^* \\ &= \frac{e^{-j(\|l_t - l_i\|_2 - \|l_t - l_k\|_2)f/c}}{(4\pi)^2\|l_t - l_i\|_2\|l_t - l_k\|_2}, \end{aligned} \tag{3.6}$$

where $(.)^*$ is the complex conjugate operator. In this way we can calculate the covariance matrix of a network of nodes observing a source. If

more than one source is present, the covariance matrix can be calculated for each source separately and then summed to find the total theoretical covariance. The only information required is the spatial coordinates of the source, and the nodes.

The spatial covariance R_S as defined by (3.6) is noise free, which is inconsistent with processing noisy audio signals in the real world. Noise can be added directly to the covariance matrix. Uncorrelated noise (perhaps thermal noise in the sensor circuitry) is independent between sensors and appears only on the diagonal of the covariance matrix. If we assume equal noise power σ_N^2 in each sensor then the uncorrelated noise covariance matrix is given by

$$R_N = \sigma_N^2 I, \tag{3.7}$$

where I is the identity matrix, which can be added to the spatial covariance to find the total covariance

$$R = R_S + R_N. \tag{3.8}$$

This is sometimes referred to as diagonal loading or Tikhonov regularization [48, 49]. Diagonal loading is useful for improving the condition number (and therefore the invertibility) of the covariance matrix, which can be a problem in practice. Diagonal loading can also help make the beamformer more robust to uncertainty in the look direction and node location.

Diffuse noise can be added in a similar way. Diffuse noise is given by [50],

$$R_D = \frac{\psi_{ik}}{\sqrt{\psi_{ii}\psi_{kk}}} = \operatorname{sinc}\left(\frac{2\pi f \|l_i - l_k\|_2}{c}\right), \tag{3.9}$$

where ψ_{ik} is the cross spectral density between the diffuse noise at node i and node k. R_D can then be added to the total covariance which now combines the spatial covariance, uncorrelated noise covariance, and dif-

fuse noise covariance,

$$R = R_S + R_N + R_D. \tag{3.10}$$

3.3 Distributed and Centralized Covariance Equivalence

It is essential to ensure that the distributed objective is equivalent to the centralized objective when formulating a distributed problem. With application to MVDR beamforming and omitting any regularization, the centralized objective is commonly of the form

$$\text{minimize} \quad w^T R w, \tag{3.11}$$

where $w \in \mathbb{R}^{|\mathcal{V}|}, R \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ and $\mathcal{V}$ is the set of all network nodes. A common choice for the distributed objective is of the form

$$\text{minimize} \quad \sum_{i\in\mathcal{V}} w_i^T R_i w_i, \tag{3.12}$$

where $w_i \in \mathbb{R}^{|\mathcal{N}_i|}, R_i \in \mathbb{R}^{|\mathcal{N}_i|\times|\mathcal{N}_i|}$ and $\mathcal{N}_i$ is the set of neighbors of node i. Equations (3.11) and (3.12) must be equivalent, which is not guaranteed. Using the distributed beamformer from [34], each local weight w_i and covariance R_i are kept at node i, and relate to node i's neighborhood. To show how non-equivalence can arise, we compare the centralized and distributed objective for the network shown in figure 3.2. The network has a

Figure 3.2: Example sensor network.

global covariance matrix R, and three local covariance matrices R_1, R_2, R_3,

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix},$$

$$R_1 = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix}, \quad R_2 = R, \quad R_3 = \begin{bmatrix} r_{22} & r_{23} \\ r_{32} & r_{33} \end{bmatrix}. \tag{3.13}$$

It also has a global weights vector w, and three local weights vectors w_1, w_2, w_3. Assume that consensus has been achieved in the weights so that the related components are equal across the global weights vector and all of the local weights vectors, and zero pad the matrices where required in order to place the weights and covariances at the correct index. Then

$$\begin{aligned} \sum_{i \in \mathcal{V}} w_i^T R_i w_i &= \begin{bmatrix} w_1 & w_2 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & 0 \\ r_{21} & r_{22} & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ 0 \end{bmatrix} \\ &\quad + \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} \\ &\quad + \begin{bmatrix} 0 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & r_{22} & r_{23} \\ 0 & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} 0 \\ w_2 \\ w_3 \end{bmatrix} \\ &= \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} 2r_{11} & 2r_{12} & r_{13} \\ 2r_{21} & 3r_{22} & 2r_{23} \\ r_{31} & 2r_{32} & 2r_{33} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} \\ &\neq w^T R w \end{aligned} \tag{3.14}$$

The distributed covariance has summed to a weighted covariance, where the weightings reflect the number of times i, j in r_{ij} appear together in a

neighbor list. The neighbor lists for the network are

$$\mathcal{N}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \ \mathcal{N}_2 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \ \mathcal{N}_3 = \begin{bmatrix} 2 \\ 3 \end{bmatrix}. \tag{3.15}$$

Let C be the scaling matrix,

$$C = \begin{bmatrix} 2 & 2 & 1 \\ 2 & 3 & 2 \\ 1 & 2 & 2 \end{bmatrix}. \tag{3.16}$$

C and the Hadamard product $(\circ)$ can be used to express the first equation in (3.14) as

$$\sum_{i \in \mathcal{V}} w_i^T R_i w_i = w^T (C \circ R) w. \tag{3.17}$$

Each component of C is found as such: consider c_{12}, count the number of times node 1 and node 2 appear in the neighbor lists (3.15) together, which is twice. The centralized and distributed problems can be made equivalent by taking the Hadamard product of the original covariance matrix with $\tilde{C}$, the componentwise inverse of C,

$$\tilde{C} = \begin{bmatrix} 1/2 & 1/2 & 1 \\ 1/2 & 1/3 & 1/2 \\ 1 & 1/2 & 1/2 \end{bmatrix}. \tag{3.18}$$

Then

$$w^T (C \circ \tilde{C} \circ R) w = w^T R w. \tag{3.19}$$

This method can be used to force equivalence between the centralized and distributed quadratic objectives. It will not work however if there are any zeros in C, which occurs any time two nodes do not share any neighbors (which is common in a realistic network). In this case we must force the

original covariance matrix to be sparse in order to achieve equivalence between the distributed and centralized beamformer expressions.

Forcing the covariance matrix to be sparse means it is no longer guaranteed positive semidefinite and therefore can not be considered a covariance matrix. An alternative approach is to generate a sparse random covariance matrix that is guaranteed positive semidefinite. For example, Matlab can generate a positive semidefinite symmetric random matrix given a certain sparsity pattern using `sprandsym()`. However, generating a random sparse covariance matrix means we no longer have information about the location of the sensors. We cannot derive the location of the sensors from the randomly generated sparse covariance matrix as the locations will be ambiguous. This is nevertheless a useful approach and was utilized in [34].

3.4 Is Bigger Better?

When considering a sensor field observing a target signal it is an interesting question to ask what is the optimal sensor distribution. Here we consider whether a small group of sensors near to the target is more effective than a large group of sensors far from the target.

Consider a single sensor. As the sensor moves away from the target, the energy from the target signal decreases as acoustic signals become attenuated with distance. The energy from the thermal noise remains constant, so moving further from the source results in lower SNR.

Now consider a uniform distribution of sensors in $\mathbb{R}^3$. The energy from the target radiates as the surface of a sphere. As we move away from the target, the SNR per sensor decreases but the radius, surface area, and number of nodes on the surface of the sphere increases. Which should we prefer; a small sphere with few sensors near the target, or a large sphere with many sensors far from the target?

We want to compare the ratio of the number of sensors on our big and

small spheres, with the ratio of the SNR from our big and small spheres. The number of sensors is $M_S(r) = \rho A = \rho 4\pi r^2$, where the S stands for sphere, ρ m^{-2} is the sensor density, and $A = 4\pi r^2$ m^2 is the surface area of the sphere. Let r be the radius of a small sphere, and αr the radius of a larger sphere with $\alpha > 1$. Assume a delay and sum beamformer, with uncorrelated equal power noise at all sensors. First we find the ratio of the number of sensors on the large and small spheres,

$$\frac{M_S(\alpha r)}{M_S(r)} = \frac{\rho 4\pi(\alpha r)^2}{\rho 4\pi r^2} = \alpha^2 \tag{3.20}$$

which shows that increasing the radius of the sphere by a factor α increases the number of sensors by α^2.

In order to find the ratio of the two SNRs we require a beamforming model. Let $Y, S \in \mathbb{R}, w, X, d, N \in \mathbb{R}^N$ be the beamformer output, target signal, weights, observations, look direction and noise with capitals indicating random variables. The output power of the beamformer is

$$\begin{aligned} \mathrm{E}\big[Y^T Y\big] &= \mathrm{E}\big[(w^T X)^T (w^T X)\big] \\ &= \mathrm{E}\big[(w^T (dS + N))^T (w^T (dS + N))\big] \\ &= \mathrm{E}\big[S^2\big] d^T w w^T d + \mathrm{E}\big[N^T w w^T N\big] \end{aligned} \tag{3.21}$$

where the final step was possible because the target and noise signals are uncorrelated. The output power of the beamformer (3.21) is composed of two terms, one from the target and one from the noise. The SNR is the ratio of these two terms,

$$\mathrm{SNR} = \frac{\mathrm{E}\big[S^2\big] d^T w w^T d}{\mathrm{E}\big[N^T w w^T N\big]}. \tag{3.22}$$

If we consider that the phase has already been adjusted for, then a delay and sum beamformer has $w = \mathbf{1}$. As the noise is uncorrelated with equal variance σ_N^2 at each node, then $\mathrm{E}[NN^T] = \sigma_N^2 I$ where I is the identity

matrix. The sensors are all on the surface of a sphere so are equidistant from the source at the centre and $d = (4\pi r)^{-1}\mathbf{1}$, so the SNR simplifies to

$$\text{SNR} = \frac{\text{E}\left[S^2\right] M}{\sigma_N^2 (4\pi r)^2} = \frac{\text{E}\left[S^2\right] \rho}{\sigma_N^2 4\pi}, \tag{3.23}$$

which does not depend on the radius. The SNR for the two spheres can now be compared,

$$\frac{\text{SNR}_S(\alpha r)}{\text{SNR}_S(r)} = \frac{\left(\frac{\text{E}\left[S^2\right] \rho}{\sigma_N^2 4\pi}\right)}{\left(\frac{\text{E}\left[S^2\right] \rho}{\sigma_N^2 4\pi}\right)} = 1. \tag{3.24}$$

So increasing the radius of the sphere by α increased the number of sensors by α^2 but made no change to the SNR.

A more realistic scenario is one where the target is a talker in a room and the sensors are uniformly distributed on the ceiling. We can apply the same process as was used above to find whether it is better to use a large or small ring of sensors in a plane. Assume the target sits in the same plane as the sensors and define the ring as the area between the circle with radius r and the circle with radius $r + \delta$. Therefore, the area of the ring is (subscript R indicates ring),

$$A_R = \pi(r + \delta)^2 - \pi r^2 = \pi(2r\delta + \delta^2). \tag{3.25}$$

Following the same argument as we did for the sphere, we find the ratio of the number of sensors between the rings of small and large radius,

$$\frac{M_R(\alpha r)}{M_R(r)} = \frac{\rho\pi(2(\alpha r)\delta + \delta^2)}{\rho\pi(2r\delta + \delta^2)}. \tag{3.26}$$

Let $\delta << 1$ so that δ^2 is negligible, then

$$\frac{M_R(\alpha r)}{M_R(r)} = \frac{\rho\pi(2(\alpha r)\delta)}{\rho\pi(2r\delta)} = \alpha, \tag{3.27}$$

which shows that an increase in the radius by a factor of α results in an increase in the number of sensors by a factor α. The ratio between the SNR for rings with large and small radius are also calculated in the same way as for spheres above, resulting in

$$\frac{\text{SNR}_R(\alpha r)}{\text{SNR}_R(r)} = \frac{\left(\frac{\text{E}\left[S^2\right]\rho\pi 2\alpha r\delta}{\sigma_N^2(4\pi\alpha r)^2}\right)}{\left(\frac{\text{E}\left[S^2\right]\rho\pi 2r\delta}{\sigma_N^2(4\pi r)^2}\right)} = \frac{1}{\alpha}. \tag{3.28}$$

Increasing the radius of the sensor ring by a factor α has increased the number of sensors and decreased the SNR. We conclude that for sensors on a plane observing a signal propagating as a sphere in $\mathbb{R}^3$, it is more efficient (for example in terms of power consumption) and the estimate will have less error by using a sensor ring with a small radius.

In order to achieve this result we assumed there was no noise coherence between sensors. In practice, sensors close to each other have correlated noise (for example interference or diffuse noise) meaning that there is a limit to how small the ring of sensors can be while still making independent observations. Beamformers are not generally limited to a ring of sensors, it is more likely that a disk of sensors would be used. The calculations above do not extend to a disk as in both the sphere and the ring case we have used the fact that the distance from the source was equal across all sensors. Investigation of the disk case would make interesting future work.

The interesting thing that this section suggests is the possibility of an optimal distribution of sensors. Perhaps there are efficiency gains possible by creating a sensor disk with high density close to the target and

lower density further from the target. In some sense this is implied when the magnitude of the beamformer weights become smaller further from the target. It may be possible to save power by swapping many lowly weighted nodes far from the source for few highly weighted nodes far from the source.

3.5 Range of Support

The previous section mentioned that there is likely to be a minimum distance between two nodes if they are going to be useful in the presence of interference or diffuse noise. Here we briefly consider what those distances might be for a diffuse noise field. The coherence of diffuse noise from sources on a sphere is given by [50]

$$R_{ij} = \operatorname{sinc}(2\pi f \|l_i - l_j\|_2 / c). \tag{3.29}$$

To make computation tractable, let us simplify the coherence such that it equals zero outside the main lobe and equals one inside the main lobe,

$$R_{ij} = \begin{cases} 1, & (2\pi f \|l_i - l_j\|_2 / c) \leq \pi \\ 0, & \text{elsewhere.} \end{cases} \tag{3.30}$$

In the case of speech processing, we can assume a limited bandwidth of say 500 to $3 \times 10^3\,\mathrm{Hz}$. Let $f_1 = 500\,\mathrm{Hz}$ then the range of support of the coherence function is,

$$\|l_i - l_j\|_2 \leq \frac{c}{2f_1} = \frac{340\,\mathrm{m\,s^{-1}}}{1 \times 10^3\,\mathrm{Hz}} = 0.34\,\mathrm{m}. \tag{3.31}$$

Now let $f_2 = 3 \times 10^3\,\mathrm{Hz}$,

$$\|l_i - l_j\|_2 \leq \frac{c}{2f_2} = \frac{340\,\mathrm{m\,s^{-1}}}{6 \times 10^3\,\mathrm{Hz}} = 0.057\,\mathrm{m}. \tag{3.32}$$

So for a small bandwidth suitable for capturing speech, the range of support of the coherence function varies between $0.05\,\mathrm{m}$ and $0.35\,\mathrm{m}$. The implication is that nodes closer to each other than $0.5c/f$ m will be less effective in reducing diffuse noise at that frequency. We also see that the optimal distribution of nodes will be different for different frequencies. In practise this suggests that using a minimum spacing between nodes which is larger than the range of support of the coherence function for the lowest frequency being measured. This will ensure that each node contributes unique observations of the diffuse noise across all frequencies.

3.6 How Many Upstream Neighbors?

In the following chapter we present a model where the active set of nodes expands from the central insertion node. Each node has one downstream neighbor only where downstream means closer to the insertion node (see chapter 4). If the network has a uniform node density and is divided into equal width layers based on distance from the insertion node, then layers will contain more nodes the further they are from central insertion node. How many upstream neighbors do we expect a node to have?

Assume a density of nodes per square meter ρ. Let the maximum radius that defines layer j be r_j, which is an integer multiple of r_1, i.e. $r_3 = 3r_1, r_7 = 7r_1$ etc. Layer 0 has $r_0 = 0$ and contains only the insertion node. The area of layer j is

$$A_j = \pi\big(r_j^2 - r_{j-1}^2\big) = \pi\Big(\big(jr_1\big)^2 - \big((j-1)r_1\big)^2\Big) \tag{3.33}$$

and layer j contains $N_j = \rho A_j$ nodes. Then,

$$\begin{aligned} N_j &= \rho A_j \\ &= \rho\pi\Big(\big(jr_1\big)^2 - \big((j-1)r_1\big)^2\Big) \\ &= \rho\pi r_1^2\big(j^2 - (j-1)^2\big) \\ &= \rho\pi r_1^2(2j-1) \end{aligned} \tag{3.34}$$

Now we can compare the number of nodes in consecutive layers. If a node sits in layer j, then it must on average have

$$\frac{N_{j+1}}{N_j} = \frac{\rho\pi r_1^2(2(j+1)-1)}{\rho\pi r_1^2(2j-1)} = \frac{2j+1}{2j-1} \tag{3.35}$$

upstream neighbors. For example, a node in layer $j = 5$ will have $13/11 = 1.18$ upstream neighbors on average, whereas a node in layer $j = 1$ will have 3. Equation (3.35) does not depend on the density or the choice of r_1. Increasing the density or r_1 both result in more nodes in every layer, in the same proportions described by (3.35). This calculation does not make sense for r_0 and results in a ratio of -1 as the area of the layer is 0.

Some of these auxiliary results may be useful to anyone implementing distributed acoustic beamformers. We now move to the main result of this thesis and present our model for distributed BFOP calculation in the following chapter.

Chapter 4

Distributed Beamformer Output

The current state of the art in distributed acoustic beamforming enables optimization of the beamformer filter weights (BFFWs) over a distributed network of nodes. The beamformer output (BFOP) can then be calculated centrally by passing the weighted observations to a collection node as in [34]. Alternatively, the BFOP can be calculated in a distributed manner but is tied to the BFFW optimization in [33]. In this section we develop a new method for calculating the BFOP in a distributed manner. It is assumed that the BFFWs have already been calculated by a second algorithm such as delay and sum or sparse distributed MVDR [34].

To motivate the new method, imagine a scenario where the target signal changes periodically i.e. a meeting that shifts between talkers. When a new talker begins to speak, the BFFWs must be reoptimized. Once the optimal BFFWs have been found, there is no need to change the BFFWs unless the situation changes. In contrast, the BFOP must be calculated constantly and streamed to the user, so it is useful to have the BFFW optimization and the BFOP calculation implemented separately.

We can already compute the BFOP independently of the filter optimization, but in a centralized manner. Centralized computation of the BFOP requires that all nodes transmit their weighted observations to a central node. This transmission can be achieved either directly or by pass-

ing data via interlying nodes. The power required for direct transmission to the central node is relatively high, which is undesirable as it reduces the lifetime of the wireless battery powered sensor network. The required transmission power can be reduced by passing each weighted observation towards the central node via interlying network nodes. This also comes with a caveat; nodes towards the center will be required to pass more information than outer nodes, and this will place an upper limit on the size of the network.

We propose a scheme that takes the passing scheme and goes further. Each node makes a local estimate of the BFOP, and passes this estimate to only one downstream neighbor. No node is required to relay data from one neighbor to another. Every node in the network now passes the same amount of data, regardless of its position in the network. The upper limit on the size of the network imposed by both types of centralized calculation has been removed. The new scheme requires less transmission power than the centralized direct computation, as passing data via interlying nodes is cheaper than direct transmission. It also requires less power than passing all observations via interlying nodes as all nodes on the interior of the network pass less data.

In the proposed scheme, each node makes a local estimate of the BFOP. This estimate is a weighted sum of two components; the current node's weighted observation, and BFOP estimates from any upstream neighbors of the current node. Figure 4.1 shows an example three layer network with node 1 as the insertion node. A layer is defined by the number of hops to the insertion node. For two nodes located in layers a and b with $b > a$, b is upstream of a, and a is downstream of b. The insertion node is chosen for being the closest to the target source. A node in the outer layer such as node 3, makes a BFOP estimate based only on its weighted observation. It uses location information to choose the cheapest downstream neighbor to pass its BFOP estimate to, which in this case is node 2. Node 2 makes a BFOP estimate by combining its own weighted observation with the BFOP

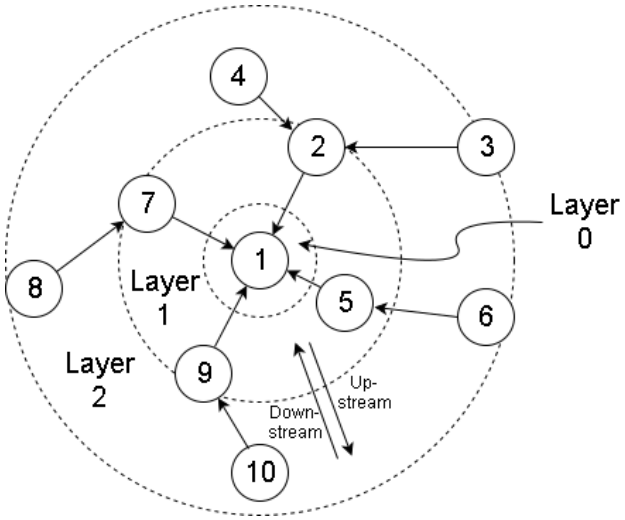

Figure 4.1: Small example sensor network. Node 1 is the insertion node. Nodes directly upstream from 1 are considered to be in layer 1. Nodes directly upstream from nodes in layer 1 are in layer 2, and so on. Nodes only ever pass a single estimate to one downstream node.

estimates of its upstream neighbors, nodes 3 and 4. Node 2 is expected to have an improved BFOP estimate over node 3 for two reasons. Firstly, node 2 is closer to the target source than node 3, and therefore has higher SNR. Secondly, node 2 can make use of three observations, while node 2 can only make use of one.

The proposed scheme imposes a tree graph over the network, which is separate to any network topology involved in the BFFW optimization. There is a vast literature on tree networks for data network routing, mobile networks, sensor networks etc. Selecting the best graph from the set of all graphs can be achieved in a number of ways. Two common approaches are shortest path and minimum hop [51, 52]. Dynamic network routing is considered in [53], and power aware routing that considers the current state of the node's energy usage and storage is discussed in [54]. [55] maximizes the lifetime of the network. Other performance criteria can also be considered, such as the overall average response time in [56]. The tree structure used in this section is chosen locally and approximately with the

aim of minimizing the transmission power, and is derived in section 4.5.

This chapter proceeds by deriving the optimal distributed combination weights that minimize the error variance of a local BFOP estimate. The performance of the proposed model is analyzed regarding the propagation of error variance and it is shown that the error variance reduces as the BFOP estimate moves towards the central insertion node. Implementation of the local combination weights is discussed followed by how to choose the downstream neighbor to approximately minimize transmission power. Simulation results are then presented that show that this distributed BFOP model trades a small decrease in performance for a large decrease in transmission power.

4.1 Optimal Combination Weights

This section introduces the system model, and finds the optimal local combination weights. Let $S \in \mathbb{R}$ be a random variable representing the Fourier coefficient of the target signal for a single bin and window index. Let $N \in \mathbb{R}^M$ be a random variable representing the Fourier coefficients of the noise across M sensors. Each node makes an observation $X = dS + N, X \in \mathbb{R}^M$ where $d \in \mathbb{R}^M$ is the look direction vector that describes the attenuation and phase shift of the target signal at each sensor location. The standard beamformer output is a weighted combination of the observations, $Z = w^T X \in \mathbb{R}$. As discussed in section 2.2.3, we can work in the real domain by assuming a mapping from complex to real numbers.

The centralized BFOP calculation only makes an estimate of the BFOP at the central node. In contrast, the proposed distributed BFOP calculation makes an estimate of the BFOP at every node. Let $Z_{new} \in \mathbb{R}$ represent the local BFOP estimate at the current node. Let $a \in \mathbb{R}^{|\mathcal{N}|}$ with the constraint that $a^T \mathbf{1} = 1$ and where $\mathcal{N}$ is the current node's neighbors, making the estimates a weighted average. We would like to find the optimal combination weights a that minimize the error variance of the BFOP estimate at

each node,

$$\begin{aligned} &\text{minimize} \quad \mathrm{E}\Big[(S - Z_{new})^2\Big] \\ &\text{subject to} \quad a^T\mathbf{1} = 1. \end{aligned} \tag{4.1}$$

Z_{new} is a weighted combination of the current node's weighted observation and BFOP estimates passed down from any upstream neighbors,

$$Z_{new} = a^T Z. \tag{4.2}$$

Let the first element of Z be $Z_1 = wX$, w is the current node's BFFW, and X is the current node's observation. The remaining elements of Z are each a BFOP estimate received from an upstream neighbor that has been passed into the current node. Each of the BFOP estimates have been found using the same Z_{new} calculation. Subbing (4.2) into (4.1),

$$\begin{aligned} \mathrm{E}\big[(S - Z_{new})^2\big] &= \mathrm{E}\big[(S - a^T Z)^2\big] \\ &= \mathrm{E}\big[(a^T(\mathbf{1}S - Z))^2\big] \\ &= \mathrm{E}\Big[\Big(\sum_{i\in\mathcal{N}} a_i(S - Z_i)\Big)^2\Big], \end{aligned} \tag{4.3}$$

where the second line was possible as $a^T\mathbf{1} = 1$ therefore $a^T\mathbf{1}S = S$. The neighborhood $\mathcal{N}$ includes the current node, and the current node's upstream neighbors. The Lagrangian is given by,

$$L(a, \nu) = \mathrm{E}\Big[\Big(\sum_{i\in\mathcal{N}} a_i(S - Z_i)\Big)^2\Big] + \nu\Big(\sum_{i\in\mathcal{N}} a_i - 1\Big). \tag{4.4}$$

Taking the gradient with respect to a_j and setting to zero,

$$\begin{aligned}\nabla_{a_j} L(a,\nu) &= 0 \\ &= 2\sum_{i\in\mathcal{N}} a_i \mathrm{E}[(S-Z_i)(S-Z_j)] + \nu \\ &= 2\sum_{i\in\mathcal{N}} a_i b_i^{(j)} + \nu \\ &= 2a^T b^{(j)} + \nu, \end{aligned} \tag{4.5}$$

where $b^{(j)} \in \mathbb{R}^{|\mathcal{N}|}$ has elements $b_i^{(j)} = \mathrm{E}[(S-Z_i)(S-Z_j)], i \in [1,\ldots,|\mathcal{N}|]$. (4.5) can be expressed over all j by introducing the matrix B, where element i,j of B is given by $\mathrm{E}[(S-Z_i)(S-Z_j)]$. Then the gradient of the Lagrangian with respect to a is,

$$\begin{aligned}\nabla_{a} L(a,\nu) &= 0 \\ &= B\tilde{a} + \frac{\nu}{2}\mathbf{1} \\ \therefore \quad \tilde{a} &= \frac{-\tilde{\nu}}{2} B^{-1}\mathbf{1}. \end{aligned} \tag{4.6}$$

Equation (4.6) must still meet the constraint,

$$\begin{aligned}\tilde{a}^T\mathbf{1} &= 1 \\ \therefore\ \tilde{\nu} &= \frac{-2}{\mathbf{1}^T B^{-1}\mathbf{1}}. \end{aligned} \tag{4.7}$$

Subbing $\tilde{\nu}$ back into $\tilde{a}$ gives us the optimal combination weights as a function of B^{-1},

$$\tilde{a} = \frac{B^{-1}\mathbf{1}}{\mathbf{1}^T B^{-1}\mathbf{1}}, \tag{4.8}$$

which depends only on the expected error at the current node, and the expected error at the upstream neighbors. This optimal combination weight vector can be used to combine the current node's weighted observation

with the estimates of any upstream neighbors, resulting in the optimal Z_{new}, the current node's estimate of the BFOP. Z_{new} is then passed to the current node's downstream neighbor. The resemblance of the optimal combination weights to the optimal MVDR weights is superficial and the two should not be confused.

4.2 Error Variance

It is desirable to have some understanding about the error variance at different parts of the network. It is expected that a downstream node will have lower error variance than its upstream neighbors. There are two reasons for this expectation, the downstream node is closer to the target so receives more target signal energy, and the downstream node makes an estimate based on a greater number of observations. Here we show that use of the optimal combination weights results in an error variance less than the smallest error variance at the upstream neighbors for a scenario with a single target source and independent noise.

Let $\mathcal{N}$ be the current node's neighborhood which includes its upstream neighbors and its own weighted observation. Let $a \in \mathbb{R}^{|\mathcal{N}|}$ be the combination weights, $Z \in \mathbb{R}^{|\mathcal{N}|}$ is a random variable representing the neighborhood estimates, $B = \mathrm{E}[bb^T] \in \mathbb{R}^{|\mathcal{N}|\times|\mathcal{N}|}$ is the covariance of the estimate errors, $b = (\mathbf{1}S - Z) \in \mathbb{R}^{|\mathcal{N}|}$, $\mathbf{1} \in \mathbb{R}^{|\mathcal{N}|}$ is a vector of ones, and $S \in \mathbb{R}$ is a random variable representing the target signal. $Z_{new} = a^T Z$ is the current node's estimate of the BFOP which is a weighted sum of the current node's weighted observation and any BFOP estimates passed from upstream neighbors. Assume a single target source corrupted by independent noise and MVDR weights which results in uncorrelated cross error, i.e. diagonal B. This allows us to decompose the error variance of Z_{new}

into a weighted sum of error variances at the upstream neighbors,

$$\begin{aligned}\mathrm{E}\big[(S - Z_{new})^2\big] &= \mathrm{E}\big[(S - a^T Z)^2\big] \\ &= \mathrm{E}\Big[\sum_{i\in\mathcal{N}} a_i^2 (S - Z_i)^2\Big] \\ &= \mathrm{E}\Big[\sum_{i\in\mathcal{N}} a_i^2 b_i^2\Big], \end{aligned} \tag{4.9}$$

where $\mathrm{E}[b_i^2] = \mathrm{E}[(S - Z_i)^2]$. Recall that the error variance of Z_{new} is minimized by the combination weights,

$$a = \frac{B^{-1}\mathbf{1}}{\mathbf{1}^T B^{-1}\mathbf{1}}. \tag{4.10}$$

As the covariance matrix $B = \mathrm{E}[bb^T]$ is diagonal with element $B_{ii} = \mathrm{E}[b_i^2]$, the inverse can be formed by taking the componentwise inverse of the diagonal elements, $B_{ii}^{-1} = (\mathrm{E}[b_i^2])^{-1}$. Therefore element a_i is given by

$$a_i = \frac{\big(\mathrm{E}[b_i^2]\big)^{-1}}{\sum_{j\in\mathcal{N}} \big(\mathrm{E}[b_j^2]\big)^{-1}}. \tag{4.11}$$

We would like to show that the error variance of Z_{new} is never greater than the smallest error variance of the upstream estimates $Z_i, \forall i \in \mathcal{N}$. Let m be the index of the upstream neighbor with the smallest error variance,

$$\mathrm{E}[b_m^2] \leq \mathrm{E}[b_i^2],\ \forall i. \tag{4.12}$$

We aim to show that

$$\mathrm{E}\Big[\sum_{i\in\mathcal{N}} a_i^2 b_i^2\Big] \leq \mathrm{E}[b_m^2]. \tag{4.13}$$

Subbing (4.11) into (4.13) and pulling out $\mathrm{E}[b_m^2]$ from the sum,

$$\mathrm{E}\Big[\sum_{i\in\mathcal{N}} a_i^2 b_i^2\Big] \leq \mathrm{E}[b_m^2] \tag{4.14}$$

$$\mathrm{E}\Big[\sum_{i\in\mathcal{N}} \left(\frac{\left(\mathrm{E}[b_i^2]\right)^{-1}}{\sum_{j\in\mathcal{N}}\left(\mathrm{E}[b_j^2]\right)^{-1}}\right)^2 b_i^2\Big] \leq \mathrm{E}[b_m^2]$$

$$\sum_{i\in\mathcal{N}} \left(\frac{\left(\mathrm{E}[b_i^2]\right)^{-1}}{\sum_{j\in\mathcal{N}}\left(\mathrm{E}[b_j^2]\right)^{-1}}\right)^2 \mathrm{E}[b_i^2] \leq \mathrm{E}[b_m^2]$$

$$\frac{\sum_{i\in\mathcal{N}}\left(\mathrm{E}[b_i^2]\right)^{-1}}{\left(\sum_{j\in\mathcal{N}}\left(\mathrm{E}[b_j^2]\right)^{-1}\right)^2} \leq \mathrm{E}[b_m^2]$$

$$1 \leq \mathrm{E}[b_m^2]\sum_{j\in\mathcal{N}}\left(\mathrm{E}[b_j^2]\right)^{-1}$$

$$1 \leq \mathrm{E}[b_m^2]\left(\left(\mathrm{E}[b_m^2]\right)^{-1} + \sum_{\{j\in\mathcal{N}|j\neq m\}}\left(\mathrm{E}[b_j^2]\right)^{-1}\right)$$

$$1 \leq 1 + \mathrm{E}[b_m^2]\sum_{\{j\in\mathcal{N}|j\neq m\}}\left(\mathrm{E}[b_j^2]\right)^{-1}, \tag{4.15}$$

which must be true as $\mathrm{E}[b_j^2], \mathrm{E}[b_m^2] \geq 0, \forall (j,m) \in \mathcal{N}$. Therefore, as equation (4.15) is true, equation (4.14) is also true, and the error variance of Z_{new} is guaranteed to be no more than the smallest error variance of $Z_i, \forall i \in \mathcal{N}$, under independent noise. The error variance decreases as the output calculation moves towards the insertion node.

Note that the performance of the distributed BFOP calculation is not guaranteed to equal or better the performance of the centralized calculation. So what is the advantage? Calculating the BFOP using the distributed combination weights approach significantly reduces the communication power required. Given our context is wireless sensors, any power saving is desirable as it results in a longer life network. The following section describes the calculation and passing of BFOP estimate error variances in practice.

4.3 Error Variance In Practise

The distributed BFOP calculation using the optimal combination weights relies on knowing the error variances in the neighborhood. How does each node calculate its own error variance?

Consider a single target source with independent noise at each node. Also assume that the noise variance is equal for all nodes, which is reasonable if the nodes are constructed identically. All error originates from error in the observations. There are two types of node, an outer layer node, and a node with upstream neighbors. An outer layer node only has error in its observation. A node with upstream neighbors has error in its observation, and error in the estimates passed into it by its upstream neighbors. Each node must pass an estimate of its error variance to its downstream neighbor so that the downstream neighbor can properly weight the estimate. We now derive the general error variance for all nodes and then look at the special case of outer layer nodes.

As in the previous section, $Z_{new} = a^T Z$ is the current node's estimate of the target signal which is a weighted combination of the current node's already weighted observation and the estimates of any upstream neighbors. These are concatenated into $Z \in \mathbb{R}^{|\mathcal{N}|}$, where $\mathcal{N}$ is the current node's neighborhood including itself, and $|\mathcal{N}|$ is the number of nodes in the neighborhood. The error variance of the current node is again given by (4.9), which indicates that the current node can calculate its own error variance by weighting and summing the error variances from the nodes in its neighborhood including itself. The only source of error variance is the observations so all that is needed is to find the error variance of each node's observation and then propagate these errors via the weighted sum of (4.9). Assume that the current node's weighted observation is element

1 of Z, i.e. $Z_1 = wX$. Then the error variance of Z_1 is

$$\begin{aligned}
\mathrm{E}\big[(S - Z_1)^2\big] &= \mathrm{E}\big[(S - wX)^2\big] \\
&= \mathrm{E}\big[(S - w(dS + N))^2\big] \\
&= \mathrm{E}\big[S^2 - 2Sw(dS + N) + (w(dS + N))^2\big] \\
&= \mathrm{E}\big[S^2 - 2S^2wd - 2SwN + w^2d^2S^2 + 2dSN + w^2N^2\big] \\
&= \mathrm{E}\big[S^2\big](wd - 1)^2 + w^2\mathrm{E}\big[N^2\big]. \qquad (4.16)
\end{aligned}$$

The total error variance can be calculated as,

$$\begin{aligned}
\mathrm{E}\bigg[\sum_{i\in\mathcal{N}} a_i^2(S - Z_i)^2\bigg] = a_1^2\Big(\mathrm{E}\big[S^2\big](wd - 1)^2 + w^2\mathrm{E}\big[N^2\big]\Big) \\
+ \sum_{\{i\in\mathcal{N}|i\neq 1\}} a_i^2\mathrm{E}\big[(S - Z_i)^2\big], \qquad (4.17)
\end{aligned}$$

where $\mathrm{E}\big[(S - Z_i)^2\big], \{i \in \mathcal{N}|i \neq 1\}$ are all passed to the current node from its upstream neighbors. For a node on the outer layer,

$$a = \frac{B^{-1}\mathbf{1}}{\mathbf{1}^T B^{-1}\mathbf{1}} = \frac{\Big(\mathrm{E}\big[(S - Z_1)^2\big]\Big)^{-1}}{\Big(\mathrm{E}\big[(S - Z_1)^2\big]\Big)^{-1}} = 1, \qquad (4.18)$$

so (4.17) becomes

$$\begin{aligned}
\mathrm{E}\bigg[\sum_{i\in\mathcal{N}} a_i^2(S - Z_i)^2\bigg] &= \mathrm{E}\big[(S - Z_1)^2\big] \\
&= \mathrm{E}\big[S^2\big](wd - 1)^2 + w^2\mathrm{E}\big[N^2\big]. \qquad (4.19)
\end{aligned}$$

The distributed BFOP can now be simulated. The calculation requires the target signal variance, the noise variance, and the assumption of independent noise. Note that for MVDR weights $wd = 1$ as required by the constraint so the variance of the target signal can be omitted.

We still require a method for computing the total transmission power.

The next section describes calculation of the total transmission power, and selection of the cheapest downstream neighbor for the current node to pass its BFOP estimate.

4.4 Linear Array Transmission Power: Direct vs Multihop

The advantage of our distributed beamformer output calculation is a reduction in the required transmission power, and therefore an extended network lifetime. It is cheaper to pass a value inwards using a multihop transmission scheme than directly. This section highlights this with an example on a linear array shown in figure 4.2.

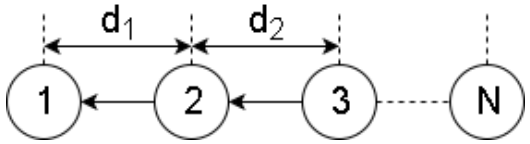

Figure 4.2: Linear node array. Less power is required to transmit from N to 1 via the interlying nodes, than to transmit directly from N to 1.

Assume each node requires a receive power of P to guarantee error free reception. The absolute distance between two nodes i and $i+1$ is $d_i \geq 0, \forall i$, and the power required to transmit from node $i+1$ to i is $P_i = Pd_i^2$, so that the power received at node i is $Pd_i^2 d_i^{-2} = P$, where we have assumed the radio power decays as d^{-2}.

The example array is linear, meaning that the direct transmission distance from N to 1 is $d_{N\to 1} = \sum_{i=1}^{N-1} d_i$. We would like to show that the power required to transmit from N to 1 via the interlying nodes (multihop) is less than the power required to transmit from N to 1 (direct). The

total transmission power required for direct transmission is given by,

$$P_{N\to 1} = P d_{N\to 1}^2 = P\left(\sum_{i=1}^{N-1} d_i\right)^2, \tag{4.20}$$

and for multihop by,

$$\sum_{i=1}^{N-1} P_i = P\sum_{i=1}^{N-1} d_i^2. \tag{4.21}$$

Inspection of (4.20) and (4.21) reveals that the direct total transmission power requirement is always greater than for multi hop, as expected. The linear array case is a simplification, a real network will be passing between nodes in a non-linear configuration. It is possible to imagine configurations where the multihop approach is expected to be more expensive, for example, if node 1 and node 2 were to swap positions in figure 4.2. The next section looks at the conditions on multihop being cheaper.

4.5 Non-Linear Array Transmission Power: Direct vs Multihop

Multihop transmission was shown to be cheaper than direct transmission for a linear array in the previous section. Real array geometries are likely to be non-linear so it is desirable to know when it is cheaper to use multihop, and when it is cheaper to use direct transmission. These conditions are derived in this section.

Consider the non-linear array shown in figure 4.3. Node 1 is the insertion node, and we want to send node N's estimate to node 1. We have two options, transmit from N to 1 via the interlying nodes (multi-hop), or directly from N to 1 (direct). Let P be a constant that represents the receive power required at all nodes for error free reception. The location of node

i is given by $l_i \in \mathbb{R}^3$, and the distance between two nodes $i+1$ and i by $\|l_{i+1} - l_i\|_2$. If we assume that the radio signals decay as an inverse square, then the power required to transmit from node $i+1$ to node i is

$$P_i = P\|l_{i+1} - l_i\|_2^2. \tag{4.22}$$

The total power for the multihop scheme is the sum of the transmission power for each single hop,

$$P_{multihop} = P\sum_{i=1}^{N-1} \|l_{i+1} - l_i\|_2^2. \tag{4.23}$$

The total power for the single hop scheme is

$$P_{singlehop} = P\|l_N - l_1\|_2^2. \tag{4.24}$$

Notice that

$$\begin{aligned}
P_{singlehop} &= P\|l_N - l_1\|_2^2 \\
&= P\left\| \sum_{i=1}^{N-1} l_{i+1} - l_i \right\|_2^2 \\
&= P\left(\sum_{i=1}^{N-1} l_{i+1} - l_i \right)^T \left(\sum_{i=1}^{N-1} l_{i+1} - l_i \right) \\
&= P\left(\sum_{i=1}^{N-1} \|l_{i+1} - l_i\|_2^2 \right) + 2P\sum_{i=1}^{N-1}\sum_{j=1, j\neq i}^{N-1} (l_{i+1} - l_i)^T(l_{j+1} - l_j) \\
&= P_{multihop} + 2P\sum_{i=1}^{N-1}\sum_{j=1, j\neq i}^{N-1} (l_{i+1} - l_i)^T(l_{j+1} - l_j)
\end{aligned} \tag{4.25}$$

from which we can conclude the following relationship,

$$\begin{aligned} P_{singlehop} &> P_{multihop} \text{ for } g > 0 \\ P_{singlehop} &= P_{multihop} \text{ for } g = 0 \\ P_{singlehop} &< P_{multihop} \text{ for } g < 0, \end{aligned} \tag{4.26}$$

where

$$g = 2P \sum_{i=1}^{N-1} \sum_{j=1, j\neq i}^{N-1} (l_{i+1} - l_i)^T (l_{j+1} - l_j). \tag{4.27}$$

The conclusion with respect to our application is that if the sum of inner products of the cross terms in g is positive, then multihop transmission will be cheaper than single hop, when we have assumed that the radio transmission loss is described by an inverse square with no overheads.

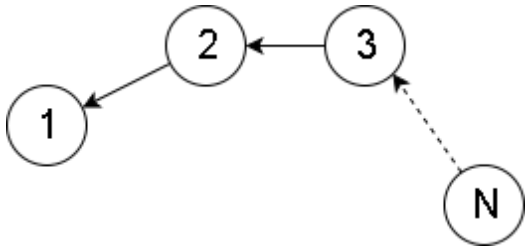

Figure 4.3: Non-linear network. When is it cheaper to transmit directly from node N to node 1?

The optimal set of paths could be found in order to define the network tree by minimizing (4.25) over all possible paths. In practice however, a node must choose its downstream neighbor without knowing the location of all of the nodes in the network. In simulation we have used a simplified criteria for choosing the cheapest downstream neighbor. The current node i selects the closest neighbor j whose location vector satisfies $(l_i - l_j)^T (l_j - l_1) \geq 0$ where l_1 is the insertion node. The location of the insertion node is not explicitly known by a node, so the location of the target is used to estimate the location of the insertion node. It is also possible

for the selected downstream neighbor to choose a downstream neighbor which is not optimal for the current node. So this method of choosing the downstream neighbor is in practice not ideal and we would expect to see some additional reduction of the required transmission power if the optimal set of downstream neighbors were used instead.

We are now able to find the optimal combination weights and have a non-optimal approximation for choosing downstream neighbors and therefore defining the tree structure. The next section presents simulation results showing that the distributed BFOP calculation leads to a slight drop in SNR performance with a large drop in transmission power requirements.

4.6 Simulation Results

This section presents simulation results confirming the desirable behaviour of our proposed distributed BFOP calculation. Two scenarios are considered, fixed area and fixed density. A target source is placed below a planar array of sensor nodes, mimicking a talker in a room with a sensor array on the ceiling. Figure 4.4 shows a typical layout. The node distribution was generated randomly for each iteration of the simulation, and each data point plotted represents the average of 10 simulation runs. The speed of sound is assumed to be $343\,\mathrm{m\,s^{-1}}$, and only direct path sound is considered. A $1\,\mathrm{s}, 48\,\mathrm{kHz}$ .flac audio file containing speech was passed through a STFT using a square root Hann window with 50% overlap.

4.6.1 Fixed Area, Increasing Density

Figure 4.5 shows the results for simulation with fixed area. At each iteration a new sensor distribution is randomly and uniformly generated over a constant area. Figure 4.5a shows four curves. The highest performance was achieved by the centralized output calculation as expected.

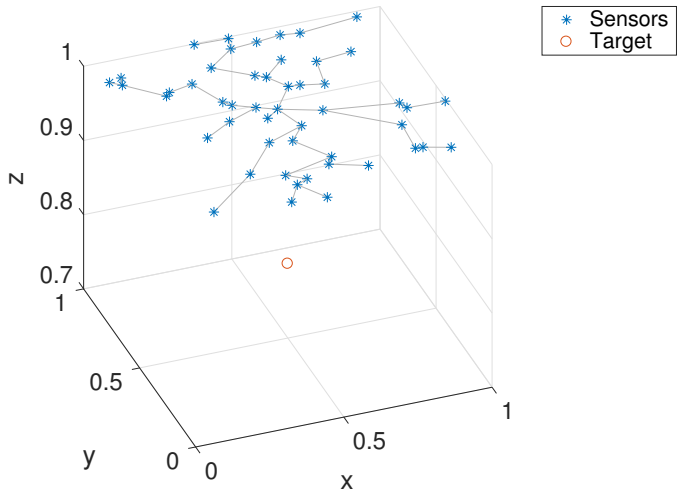

Figure 4.4: An example simulation random layout with 50 sensor nodes. The sensor nodes are placed on a plane above the source, imitating a talker in a room with nodes installed on the ceiling. The connection from each node to its downstream neighbor is also shown in grey, as used in the distributed BFOP calculation.

The optimal distributed BFOP calculation performed slightly worse than the centralized calculation. The distributed mean curve represents a distributed BFOP calculation where rather than the optimal local combination weights, each node simply averages its inputs. This results in reduced SNR performance, as we would expect. The last curve gives the performance of the sensor closest to the target source for reference.

Figure 4.5b shows that after the network grows larger than 5 sensors, the distributed total power requirement becomes constant (see section 4.5 for the transmission power model, relative transmission power refers to the fact that we have assumed a minimum receive power of 1 for both the distributed and centralized cases). This may at first seem counterintuitive however each added node can just as easily reduce the total power requirement as increase it. For example, a power reduction would result from a new node being placed between an existing node and the insertion node. This behaviour occurs because we have fixed the network area and

varied the node density.

Comparing the total power requirement for the distributed and centralized cases we can see that the distributed total power requirement is always less than or equal to the centralized case, and for large networks the distributed computation is far cheaper. The distributed total power requirement appears to be constant for networks larger than 5 nodes, while the centralized total power requirement appears to be linear. This seems reasonable, as adding a node to the distributed case could increase or decrease the power requirement, whereas adding a node to the centralized case always increases the power requirement. Furthermore, the expected distance between a node and the insertion node is constant, so the expected power required per node in the centralized case is also constant.

These results suggest that the distributed performance can be improved by simply adding nodes without increasing the total power requirement. This is surprising, and a result of some of our assumptions and omissions. For example, we have not considered computation overheads which would surely increase the power requirement as nodes are added. An upper limit on the density would also be imposed by the fact that real sensors take up space.

The third plot figure 4.5c contains no new information but compares the SNR vs total power requirement for the distributed and centralized BFOP calculations. It is clearly cheaper to achieve a certain SNR by using a distributed calculation with more nodes than to use the centralized calculation, though the centralized calculation has a higher maximum SNR for a fixed network size.

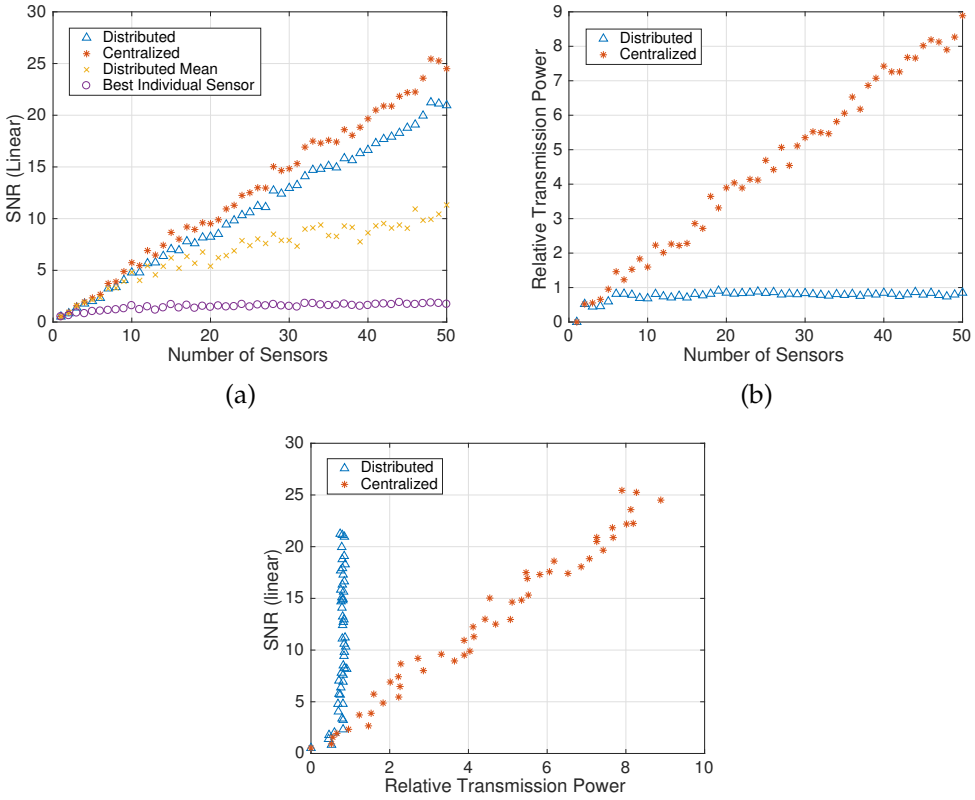

Figure 4.5: Simulation results for fixed area. Nodes are added by increasing the density. (a) The centralized BFOP calculation has the highest performance, with the distributed BFOP calculation slightly lower. The distributed mean curve represents a distributed calculation where each node averages its inputs rather than using the optimal combination weights. The best individual sensor gives the performance of the sensor nearest the target source. (b) The distributed BFOP calculation requires far less power for transmission. In a real installation this would allow the battery powered sensor network to remain operational for longer by using the distributed BFOP approach. (c) Comparison of SNR vs power for the distributed and centralized BFOP calculation. It is cheaper to achieve a specific SNR by using the distributed BFOP calculation.

4.6.2 Fixed Density, Increasing Area

This section presents simulation results with the node density held constant. A full network of 50 nodes was randomly and uniformly distributed, from which the active set of nodes was chosen. So for a network of 20 nodes, the 20 closest nodes to the source are chosen from the uniformly distributed 50. In this way, the area increases on average as more nodes are included. The increase in performance as nodes are added is expected to get smaller as the newly added nodes are further from the target source and have lower observation SNR. This is in contrast with the previous section where each additional node could be anywhere in the sensor plane.

Similar to the fixed area results above, figure 4.6a shows the centralized output calculation outperforming the distributed calculation. Again the distributed mean and best individual sensor results are included for reference.

The total power results shown in figure 4.6b show that the distributed BFOP calculation is far cheaper than the centralized BFOP calculation. The distributed calculation appears to be linear while the centralized calculation appears to be exponential. Therefore the power savings available by using the distributed calculation become larger for larger networks.

Figure 4.6c compares the SNR vs total power for the distributed and centralized BFOP calculation. Again, it is clearly cheaper to achieve a target SNR using the distributed calculation, though the maximum possible SNR on a fixed size network is still higher for the centralized BFOP calculation.

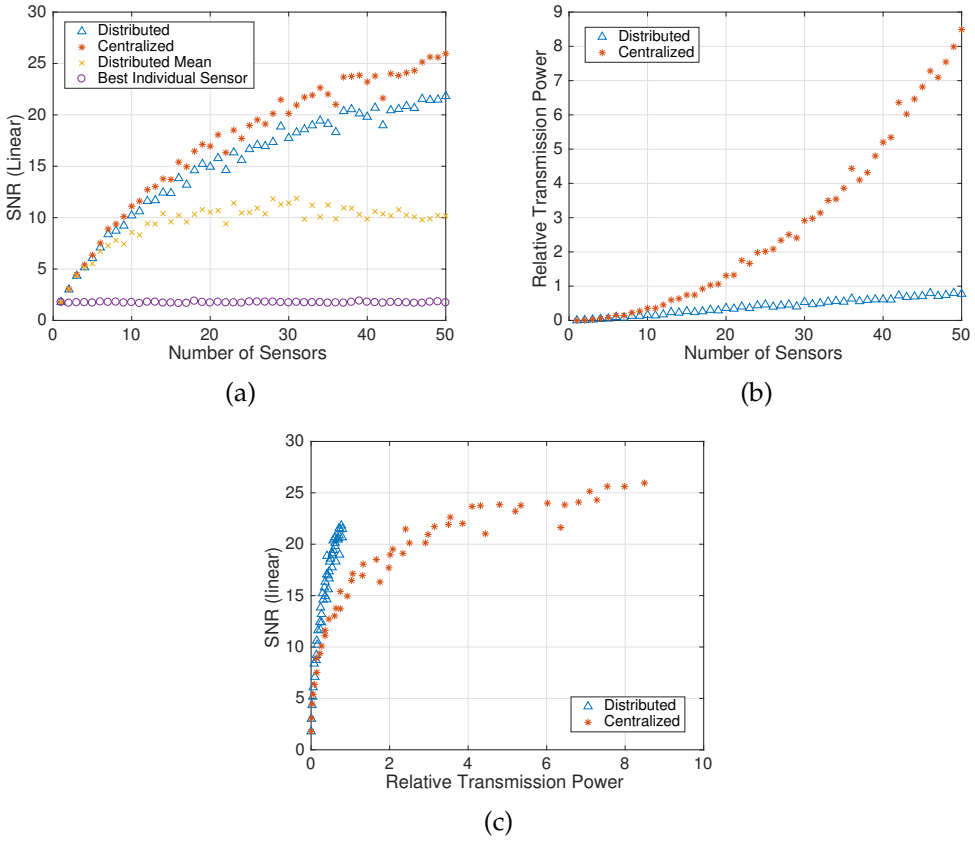

Figure 4.6: Simulation results for fixed density. Nodes are added by including the next closest node in the network, effectively increasing the network area. (a) The centralized BFOP calculation gives the highest performance per node. The distributed computation has a slightly worse SNR per node performance. (b) Distributed BFOP calculation is far cheaper than centralized BFOP calculation. This effect becomes more pronounced the larger the network. (c) Comparison of SNR vs power for the distributed and centralized BFOP calculations. It is cheaper to achieve a certain SNR using the distributed calculation.

4.7 Discussion and Future Work

The simulation results above indicate that calculating the BFOP in a distributed manner requires less total transmission power than a centralized computation. For a fixed SNR, it is cheaper to use distributed BFOP calculation with extra nodes, than to use centralized BFOP calculation. A PSN with many extra dormant nodes available for recruitment may find that it is more efficient to extend the size of the network to save power using distributed BFOP. For a fixed network size however, the maximum SNR possible is still achieved by centralized computation.

There are some restrictions on the applicability of these results. The simulations were run for a single target source with independent noise at the sensors. Future work includes extending these results to include sources of interference and diffuse noise. Calculation of the optimal combination weights requires an assumption about the observation error variance at each node. Further investigation is required to identify the best assumptions, this step will be further complicated by the inclusion of interferers in the model. It may be found that distance from the target remains a useful basis for the error variance assumption even when MVDR BFFWs are used as they already cancel the interference.

The power calculation that was used is expected to overestimate the power requirement of the distributed BFOP calculation. This is due to the approximation of the optimal set of downstream neighbors. It would be interesting to compare the results with a network using the optimal set of downstream neighbors, to quantify the cost of the approximation.

Ideal radio transmission loss has been used throughout this thesis. Real world radio transmission losses are expected to be higher. Future work could make use of a more realistic radio loss model. It is expected that using higher radio transmission losses in the model would only exaggerate the results presented above. Higher transmission losses should have a greater detrimental effect on the centralized transmission than on the

distributed transmission. Future work could also include a more comprehensive power model that considers computation power overheads within the nodes.

Chapter 5

Conclusion

Distributed beamforming is already capable of the distributed optimization of the beamforming filter on an arbitrary network. However, the beamformer output calculation is left centralized or is tied to the filter optimization. We have proposed a new method for calculating the output of a beamformer in a distributed manner. The new method is independent of the beamforming filter optimization allowing it to run on a different time scale, and to be used with any beamforming filter optimization.

Simulation results show that the new scheme trades a small decrease in SNR performance for a large decrease in total transmission power. The results suggest it may be possible to use this method to increase the SNR while reducing transmission power, by the addition of nodes. As wireless networks are battery powered, any reduction in power use is an extension of the network lifetime.

Networks of sensors are becoming increasingly prevalent in our society. We expect this trend to extend to public sensor networks, where a central body provides a sensor network for the public use. A user is able to insert an instruction at any point and have the network calculate and return the solution. A network such as this will necessarily be unlimited in size and depend on distributed processing. The results presented in this thesis make a small contribution towards the reality of such networks.

Appendix A

A.1 Hermitian Matrix Has Real Eigenvalues

Let $A \in \mathbb{C}^{N \times N}$ be hermitian, so $A = A^H$. Let $x \in \mathbb{C}^N$ be the eigenvector of A associated with the eigenvalue λ. Then,

$$\begin{aligned} Ax &= \lambda x \\ (Ax)^H x &= (\lambda x)^H x \\ x^H A x &= \lambda^H x^H x \\ \lambda x^H x &= \lambda^H x^H x \\ \therefore\ \lambda &= \lambda^H. \end{aligned} \tag{A.1}$$

A.2 Parseval's Theorem

Energy must be preserved across domains. Parseval provided us with a relationship between energy in the time domain and energy in the frequency domain. Let $x[n], n \in [1, \ldots, N]$ be the time domain signal, and $X[k]$ be the

Fourier transform of $x[n]$, then

$$\begin{aligned}
X[k] &= \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N} \\
\Rightarrow |X[k]|^2 &= X^H[k] X[k] \\
&= \sum_{n=0}^{N-1} x^*[n] e^{j2\pi kn/N} \sum_{m=0}^{N-1} x[m] e^{-j2\pi km/N} \\
&= \sum_{n=0}^{N-1} x^*[n] \sum_{m=0}^{N-1} x[m] e^{j2\pi k(n-m)/N} \\
\Rightarrow \sum_{k=0}^{N-1} |X[k]|^2 &= \sum_{k=0}^{N-1} \sum_{n=0}^{N-1} x^*[n] \sum_{m=0}^{N-1} x[m] e^{j2\pi k(n-m)/N} \\
&= \sum_{n=0}^{N-1} x^*[n] \sum_{m=0}^{N-1} x[m] \sum_{k=0}^{N-1} e^{j2\pi k(n-m)/N}. \qquad \text{(A.2)}
\end{aligned}$$

The summation on the right hand side is a geometric series,

$$\begin{aligned}
\sum_{k=0}^{N-1} e^{j2\pi k(n-m)/N} &= \frac{e^{j2\pi k(n-m)} - 1}{e^{j2\pi k(n-m)/N} - 1} \\
&= \begin{cases} N & n = m \\ 0 & n \neq m \end{cases} \qquad \text{(A.3)}
\end{aligned}$$

so

$$\sum_{k=0}^{N-1} |X[k]|^2 = \frac{1}{N} \sum_{n=0}^{N-1} |x[n]|^2. \qquad \text{(A.4)}$$

Note that this is the convention that Matlab uses and it is also possible to achieve the preservation of energy by other placement of the scaling.

Bibliography

[1] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors," *Communications of the ACM*, vol. 43, no. 5, pp. 51–58, 2000.

[2] D. Culler, D. Estrin, and M. Srivastava, "Guest editors' introduction: Overview of sensor networks," *Computer*, vol. 37, no. 8, pp. 41–49, 2004.

[3] K. Eneman, H. Luts, J. Wouters, M. Büchler, N. Dillier, W. Dreschler, M. Froehlich, G. Grimm, V. Hohmann, R. Houben, *et al.*, "Evaluation of signal enhancement algorithms for hearing instruments," in *Signal Processing Conference, 2008 16th European*, pp. 1–5, IEEE, 2008.

[4] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the world with wireless sensor networks," in *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, vol. 4, pp. 2033–2036, IEEE, 2001.

[5] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[6] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*, vol. 23. Prentice hall Englewood Cliffs, NJ, 1989.

[7] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE/ACM Transactions on Networking (TON)*, vol. 14, no. SI, pp. 2508–2530, 2006.

[8] D. Shah *et al.*, "Gossip algorithms," *Foundations and Trends® in Networking*, vol. 3, no. 1, pp. 1–125, 2009.

[9] S. P. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers.," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[10] J. F. Mota, J. M. Xavier, P. M. Aguiar, and M. Puschel, "Distributed basis pursuit," *IEEE Transactions on Signal Processing*, vol. 60, no. 4, pp. 1942–1956, 2012.

[11] J. F. Mota, J. M. Xavier, P. M. Aguiar, and M. Puschel, "D-admm: A communication-efficient distributed algorithm for separable optimization," *IEEE Transactions on Signal Processing*, vol. 61, no. 10, pp. 2718–2723, 2013.

[12] T.-H. Chang, M. Hong, W.-C. Liao, and X. Wang, "Asynchronous distributed admm for large scale optimization part 1: Algorithm and convergence analysis," *IEEE Transactions on Signal Processing*, vol. 64, no. 12, pp. 3118–3130, 2016.

[13] T.-H. Chang, W.-C. Liao, M. Hong, and X. Wang, "Asynchronous distributed admm for large scale optimization part 2: Linear convergence analysis and numerical performance," *IEEE Transactions on Signal Processing*, vol. 64, no. 12, pp. 3131–3144, 2016.

[14] G. Zhang and R. Heusdens, "Bi-alternating direction method of multipliers.," in *ICASSP*, pp. 3317–3321, IEEE, 2013.

[15] G. Zhang, R. Heusdens, and W. B. Kleijn, "On the convergence rate of the bi-alternating direction method of multipliers.," in *ICASSP*, pp. 3869–3873, IEEE, 2014.

[16] G. Zhang and R. Heusdens, "Bi-alternating direction method of multipliers over graphs.," in *ICASSP*, pp. 3571–3575, IEEE, 2015.

[17] G. Zhang and R. Heusdens, "On simplifying the primal-dual method of multipliers," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4826–4830, IEEE, 2016.

[18] B. D. Van Veen and K. M. Buckley, "Beamforming: A versatile approach to spatial filtering," *IEEE assp magazine*, vol. 5, no. 2, pp. 4–24, 1988.

[19] S. A. Schelkunoff, "A mathematical theory of linear arrays," *Bell Labs Technical Journal*, vol. 22, no. 1, pp. 80–107, 1943.

[20] D. E. Dudgeon, "Fundamentals of digital array processing," *Proceedings of the IEEE*, vol. 65, no. 6, pp. 898–904, 1977.

[21] J. Capon, "High-resolution frequency-wavenumber spectrum analysis," *Proceedings of the IEEE*, vol. 57, no. 8, pp. 1408–1418, 1969.

[22] B. Widrow, P. Mantey, L. Griffiths, and B. Goode, "Adaptive antenna systems," *Proceedings of the IEEE*, vol. 55, no. 12, pp. 2143–2159, 1967.

[23] L. Griffiths, "A simple adaptive algorithm for real-time processing in antenna arrays," *Proceedings of the IEEE*, vol. 57, no. 10, pp. 1696–1704, 1969.

[24] O. L. Frost III, "An algorithm for linearly constrained adaptive array processing," *Proceedings of the IEEE*, vol. 60, no. 8, pp. 926–935, 1972.

[25] S. Applebaum, "Adaptive arrays," *IEEE Transactions on Antennas and Propagation*, vol. 24, no. 5, pp. 585–598, 1976.

[26] B. Widrow, "A review of adaptive antennas," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'79.*, vol. 4, pp. 273–278, IEEE, 1979.

[27] Y. Zeng and R. C. Hendriks, "Distributed Delay and Sum Beamformer for Speech Enhancement via Randomized Gossip.,"

IEEE/ACM Trans. Audio, Speech & Language Processing, vol. 22, no. 1, pp. 260–273, 2014.

[28] R. Heusdens, G. Zhang, R. C. Hendriks, Y. Zeng, and W. B. Kleijn, "Distributed MVDR beamforming for (wireless) microphone networks using message passing," in *Acoustic Signal Enhancement; Proceedings of IWAENC 2012; International Workshop on*, pp. 1–4, VDE, 2012.

[29] A. Bertrand and M. Moonen, "Distributed adaptive node-specific signal estimation in fully connected sensor networks-part 1: Sequential node updating," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5277–5291, 2010.

[30] A. Bertrand and M. Moonen, "Distributed node-specific LCMV beamforming in wireless sensor networks," *Signal Processing, IEEE Transactions on*, vol. 60, no. 1, pp. 233–246, 2012.

[31] S. Markovich-Golan, S. Gannot, and I. Cohen, "Distributed multiple constraints generalized sidelobe canceler for fully connected wireless acoustic sensor networks," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 2, pp. 343–356, 2013.

[32] M. O'Connor and W. B. Kleijn, "Diffusion-based distributed MVDR beamformer.," in *ICASSP*, pp. 810–814, IEEE, 2014.

[33] T. Sherson, W. B. Kleijn, and R. Heusdens, "A distributed algorithm for robust LCMV beamforming.," in *ICASSP*, pp. 101–105, IEEE, 2016.

[34] M. O'Connor, W. B. Kleijn, and T. D. Abhayapala, "Distributed sparse MVDR beamforming using the bi-alternating direction method of multipliers.," in *ICASSP*, pp. 106–110, IEEE, 2016.

[35] S. Gannot, D. Burshtein, and E. Weinstein, "Signal enhancement using beamforming and nonstationarity with applications to speech,"

IEEE Transactions on Signal Processing, vol. 49, no. 8, pp. 1614–1626, 2001.

[36] R. T. Rockafellar, *Convex analysis*. Princeton university press, 2015.

[37] J. L. Lagrange, *Mécanique analytique*, vol. 1. Mallet-Bachelier, 1853.

[38] P. Bussotti, "On the genesis of the lagrange multipliers," *Journal of optimization theory and applications*, vol. 117, no. 3, pp. 453–459, 2003.

[39] D. Gabor, "Theory of communication. Part 1: The analysis of information," *Electrical Engineers-Part III: Radio and Communication Engineering, Journal of the Institution of*, vol. 93, no. 26, pp. 429–441, 1946.

[40] R. Martin and R. V. Cox, "New speech enhancement techniques for low bit rate speech coding," in *Speech Coding Proceedings, 1999 IEEE Workshop on*, pp. 165–167, IEEE, 1999.

[41] T. G. Stockham, Jr., "High-speed convolution and correlation," in *Proceedings of the April 26-28, 1966, Spring Joint Computer Conference*, AFIPS '66 (Spring), (New York, NY, USA), pp. 229–233, ACM, 1966.

[42] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of computation*, vol. 19, no. 90, pp. 297–301, 1965.

[43] D. Messerschmitt, "Stationary points of a real-valued function of a complex variable," *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2006-93*, 2006.

[44] K. B. Petersen, M. S. Pedersen, *et al.*, "The matrix cookbook," *Technical University of Denmark*, vol. 7, p. 15, 2008.

[45] J. K. Au, *An ab initio approach to the inverse problem-based design of photonic bandgap devices*. PhD thesis, California Institute of Technology, 2007.

[46] M. S. Bartlett, "Smoothing periodograms from time series with continuous spectra," *Nature*, vol. 161, no. 4096, pp. 686–687, 1948.

[47] S. Haykin, "Adaptive filter theory," *2nd. ed., Prentice-Hall, Englewood Cliffs, NJ*, 1991.

[48] J. Li, P. Stoica, and Z. Wang, "On robust capon beamforming and diagonal loading," *IEEE Transactions on Signal Processing*, vol. 51, no. 7, pp. 1702–1715, 2003.

[49] T. Laseetha and R. Sukanesh, "Robust adaptive beamformers using diagonal loading," *Cyber J., Multidiscipl. J. Sci. Technol.-J. Sel. Areas Telecommun.*, vol. 2, no. 3, pp. 73–79, 2011.

[50] I. McCowan, M. Lincoln, and I. Himawan, "Microphone array shape calibration in diffuse noise fields," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 3, pp. 666–670, 2008.

[51] J.-H. Chang and L. Tassiulas, "Maximum lifetime routing in wireless sensor networks," *IEEE/ACM Transactions on networking*, vol. 12, no. 4, pp. 609–619, 2004.

[52] D. Baker and A. Ephremides, "The architectural organization of a mobile radio network via a distributed algorithm," *IEEE Transactions on communications*, vol. 29, no. 11, pp. 1694–1701, 1981.

[53] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," *Mobile computing*, pp. 153–181, 1996.

[54] J. Gomez, A. T. Campbell, M. Naghshineh, and C. Bisdikian, "Power-aware routing in wireless packet networks," in *Mobile Multimedia Communications, 1999.(MoMuC'99) 1999 IEEE International Workshop on*, pp. 380–383, IEEE, 1999.

[55] Q. Li, J. Aslam, and D. Rus, "Online power-aware routing in wireless ad-hoc networks," in *Proceedings of the 7th annual international conference on Mobile computing and networking*, pp. 97–107, ACM, 2001.

[56] J. Palmer and I. Mitrani, *Optimal tree structures for large service networks*. IEEE, 2005.